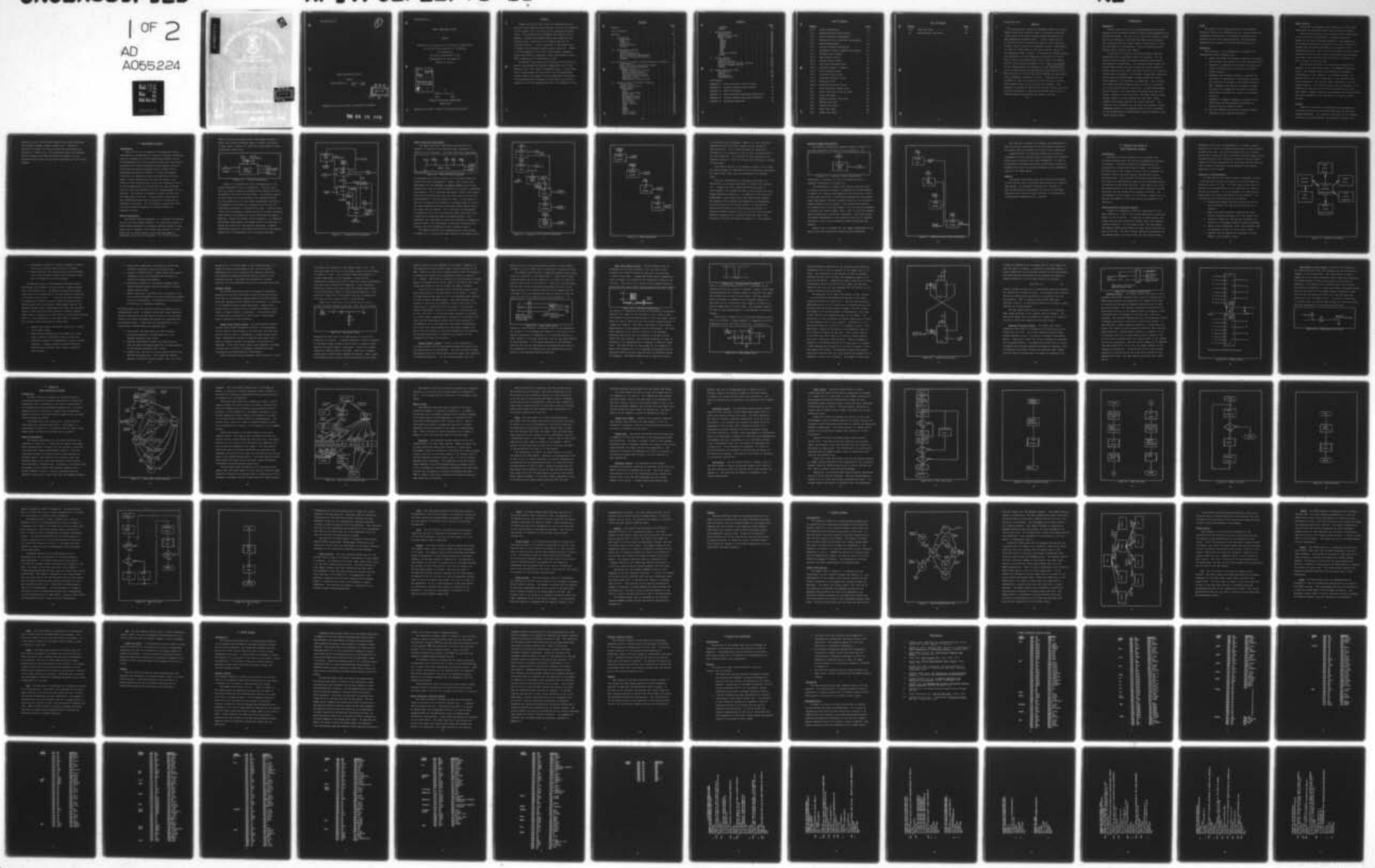
AD-A055 224 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
XEROX CONTROLLER DESIGN.(U)
MAR 78 B W CORR
UNCLASSIFIED AFIT/GE/EE/78-11 NL

1 OF 2
AD
A055224



AFIT/GE/EE/78-11

(1)

XEROX CONTROLLER DESIGN

THESIS

AFIT/GE/EE/78-11 Brian W. Corr
Capt USAF

AFIT/GE/EE/78-11

XEROX CONTROLLER DESIGN

THESIS

Presented to the Faculty of the School of Engineering
of the Air Force Institute of Technology
Air University
in Partial Fulfillment of the
Requirements for the Degree of
Master of Science

by
Brian W. Corr, B.S.
Capt USAF
Graduate Electrical Engineering

March 1978

Preface

Though the typical AFIT thesis is completed over the period of time spanning three quarters, this one was completed in two quarters due to the loss of the originally selected thesis. The driving force throughout this project was to produce a workable and usable product and though one last problem exists I feel that is not serious and will be corrected shortly. I tried to present in this paper what I thought a microcomputer design project should be. Though it is not apparent in the body of this report, shipping delays and defective components caused considerable concern about completing this undertaking on time.

My deepest thanks go to my advisor Capt Pete Miller who gave me encourgement and helpful suggestions when it most appeared that I had chosen an impossible goal. I also would like to acknowledge the trust which the AFIT Engineering Student Council has placed in me by supporting this project. The enthusiasm and willingness to help shown by my fellow students at AFIT was inspirational and I am indebted to them.

Contents

Contents

List of Figures

List of Figures

Abstract

A microprocessor system was designed, based on the Imsai 8048CC Single Board Computer (SBC), to identify authorized volume users of the Xerox copying machine operated by the AFIT Engineering Student Council. A power relay mounted on the SBC was used to enable the Xerox to make copies and a buffer circuit, based on the SN74121 one shot, was employed to count the copies made so that they could be charged to the identified user's account. An interactive FORTRAN program was implemented, on the CYBER system available at AFIT, to keep a file of authorized users, their unique user numbers, and the identification necessary to bill them. The program in the microprocessor was designed to allow the Student Council treasurer of the Student Council to set the copy cost (initially 5¢), the code used to identify valid user numbers, and any account total (increment, decrement, and set). Routines are also provided to clear all accounts (to zero) and print all non-zero account totals on the TI model 735 terminal. The program is interrupt driven and the account totals, copy cost, and code maintained in memory are protected from loss by a battery backup circuit which was added to the SBC.

I INTRODUCTION

Background

The Air Force Institute of Technology Engineering School Student Council presently operates a Xerox copying machine in the the school library. The cost to the council for making a single copy on the machine is less than five cents and by charging five cents per copy the council makes a profit which is used for the benefit of the students. There are presently three methods used to collect this charge.

A coin box is located on the machine which allows the production of one copy when a nickel is inserted. For the multiple copies needed by students working on their thesis, this method is unacceptable. An alternate "honor system" was devised which allows the user to make the desired number of copies, manually count them, and place the money in a cash box located near the machine. This system, though more convenient, has a number of disadvantages. Because copies can easily stick together, it is possible to undercount. This, coupled with users who simply forget to pay, has led to a continual six to ten percent loss of copy count. Another disadvantage of this system is that the council treasurer must sort and count a variety of coins and bills. The final method assigns account numbers to those students who want them. These students then simply sign for the copies they make. This method is very convenient for the users but requires a large bookkeeping effort by the treasurer and is still susceptible to the miscounting and forgetfullness errors described in the "honor system" method.

Scope

The scope of this undertaking will be the complete selection, design, and implementation of a system capable of performing the functions of accounting, controlling the operation of the Xerox copying mahcine, and maintaining identification records of the authorized account users.

Objectives

The objectives to be accomplished to implement an operational system, therfore, are:

1. Select and design the hardware necessary to maintain user accounts and control use of the copying machine.
2. Implement a procedure which permits the unique and secure identification of valid users and their account numbers.
3. Implement the software necessary to perform the functions of incrementing, decrementing, setting, and clearing the accounts in the Xerox Controller. Also, implement the software required to change the copy cost, change the code used to identify valid users, and print the totals in all nonzero accounts.
4. Implement a program to maintain a record of all valid users and the information necessary to identify and bill them for usage.
5. Test and validate the system to assure reliable operation of all required functions.

Sytem Overview

The system is designed as two seperate but interrelated components; the Xerox Controller and the Accounts Program.

The Xerox Controller is an interrupt driven system based on the Intel 8035 microprocessor. Keyboard inputs are recogn'zed as either treasurer commands or user numbers and the requested function is performed or the user's account is fetched from the memory depending on the position of the User/Control switch. When a valid user number has been entered the Xerox Controller will enable the copying machine and upon detecting a copy pulse from the machine will increment the user's account by the copy cost.

The Accounts Program is an interactive program which operates on the CYBER system under the INTERCOM timesharing option. The program creates and operates an a database containing up to 500 accounts. The database contains the account number, up to 30 characters of user identification, class designation, and the user number necessary to use the copying machine. The operations which can be performed on the database are selecting the key for the production of valid user numbers, adding, deleting, and changing user identification and class designations, and printing all or any one class accounts.

Outline

Chapter II is devoted to analysing the requirements of the entire system and of the Xerox Controller and Accounts Program seperately. The selection and design of the hardware, including the microprocessor, is presented in Chapter III.

Chapters IV and V describe the design of the Xerox Controller and Accounts Program software respectively. Testing is described in Chapter VI and the results and conclusions are presented in Chapter VII. The appendices contain the software source listings, The EPROM programming procedure, and the operating instructions for the Xerox Controller and the Accounts Program.

II REQUIREMENTS ANALYSIS

Introduction

This section will be devoted to analysing the system requirements as determined in a number of informal conferences with the treasurer of the AFIT Engineering Student Council. The requirements will first be given verbally for the overall system and will then be presented using diagrams similar to the Structured Analysis Design Technique (SADT) (Ref 9) terminology developed by Softech. In this terminology, activities are modeled as boxes with descriptive function names. Arrows entering from the left are inputs to the function, those entering from the top are considered control on the accomplishment of the function, and those entering from the bottom are mechanisms for the accomplishment of the function. The outputs are modeled by arrows exiting from the right side of the box. A more detailed explanation of this modeling procedure can be obtained by consulting the reference cited above. The two parallel systems into which the overall system decomposes are further defined in separate sections.

System Requirements

The primary system requirement is to maintain the accounting for the Xerox copying machine. The user must be able to input enough information to uniquely identify himself so that copies can be charged to his account upon receipt of a copy pulse from the Xerox machine. He will also remit payment on receiving a bill from the treasurer. The outputs required

from the system are account bills, user numbers issued to users, and a machine enabling signal to permit valid users to make copies. Figure II-1 shows the relationship of these requirements graphically.

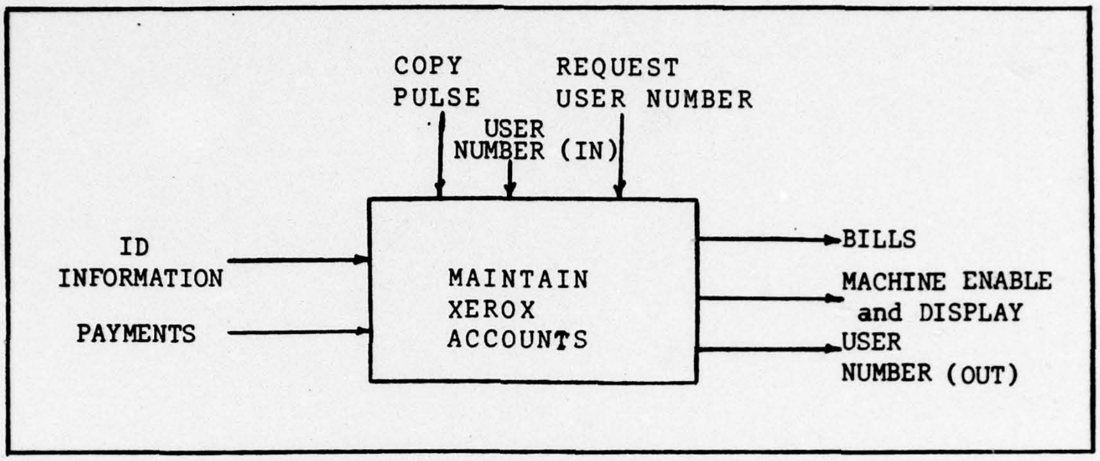


Figure II-1. System Requirements

Figure II-2 indicates the first decomposition of the system requirements into its components. Activity boxes 2 and 3 are manual treasurer operations which handle the direct interface of the users with the system. The actions required by these boxes will be described in the operating manuals for the system and if all the other requirements indicated on the diagram are satisfied the treasurer will have all the information necessary to accomplish the tasks. Activity boxes 1 and 4 are very diverse in nature and have no direct connection which would be indicated by an arrow connecting the boxes in question. Though the purpose of this analysis is to formalize and specify requirements, the design decision is made here to divide the system into two parallel subsystems. Separate requirements analyses of the Xerox Controller and the Accounts Program will be accomplished in the following sections.

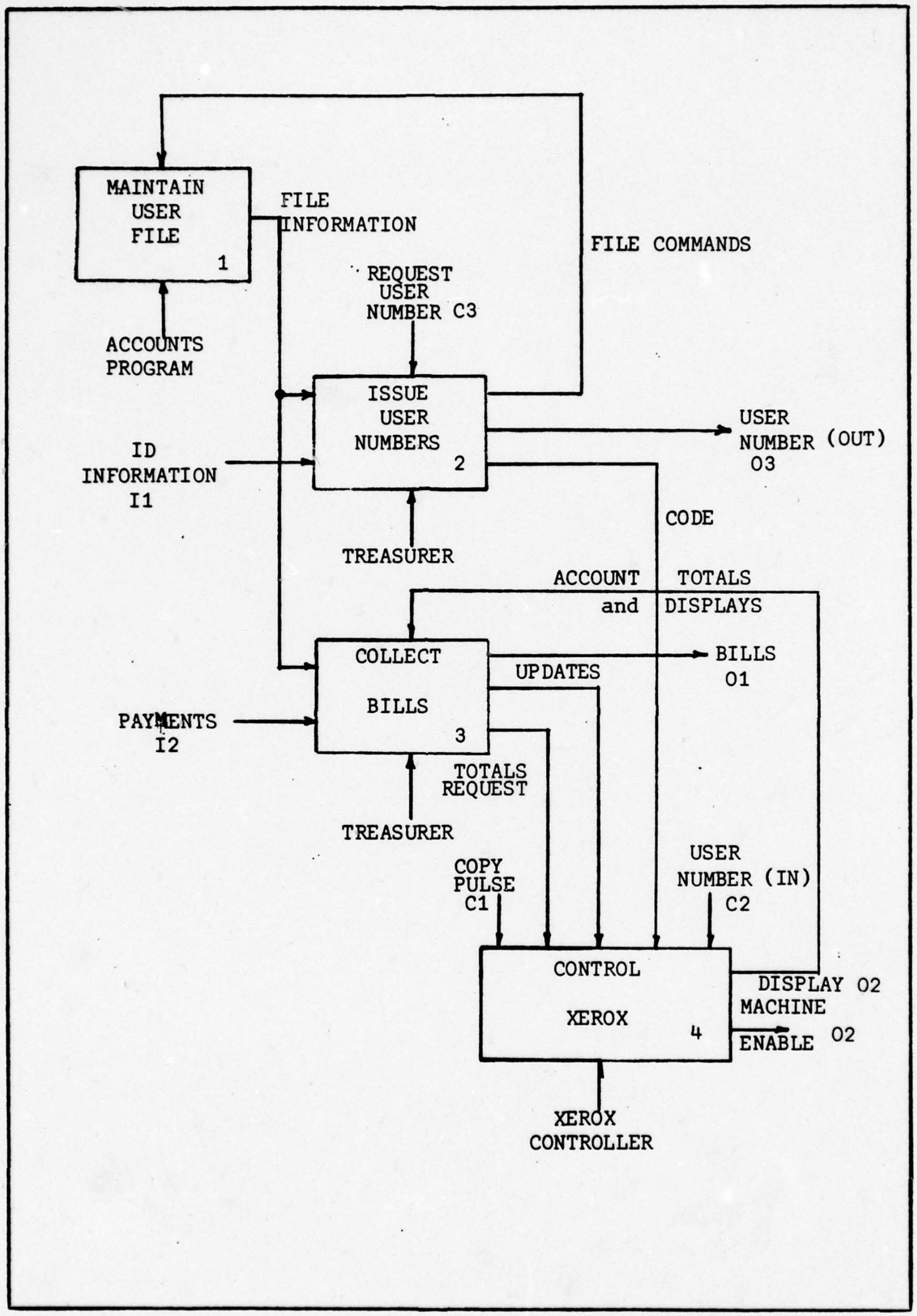


Figure II-2. Decomposed System Requirements

Xerox Controller Requirements

The Xerox Controller requirements derived from the analysis of Figure II-2 are depicted again seperately in Figure II-3 for reference. The Xerox Controller requirements

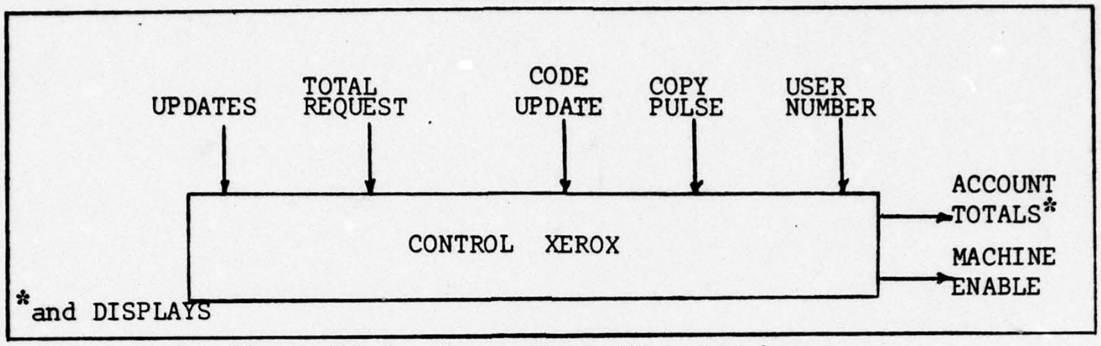


Figure II-3. Xerox Controller Requirements

are decomposed as shown in Figure II-4 to further refine and specify the exact requirements. Boxes 1 and 2 depict the requirement for the controller to recognize commands to set the cost charged for a copy and the decoding key to be used in validating user numbers. Box 3, Validate User Number, takes a user number entered by the prospective user and under control of the code must determine if the user number is valid. If the number is valid the controller must produce an enabling signal to send to the Xerox copying machine and an account number against which the copies will be charged. Box 4 indicates the requirement to charge the copy cost to the account number from box 3 when a copy pulse is detected from the copying machine. Box 5 represents the requirement to allow the treasurer to change the account totals. The requirement to produce a listing of the accounts and their respective totals is shown in box 6.

The Update Accounts box is decomposed one level further in Figure II-5 to specify the exact nature of the updates which

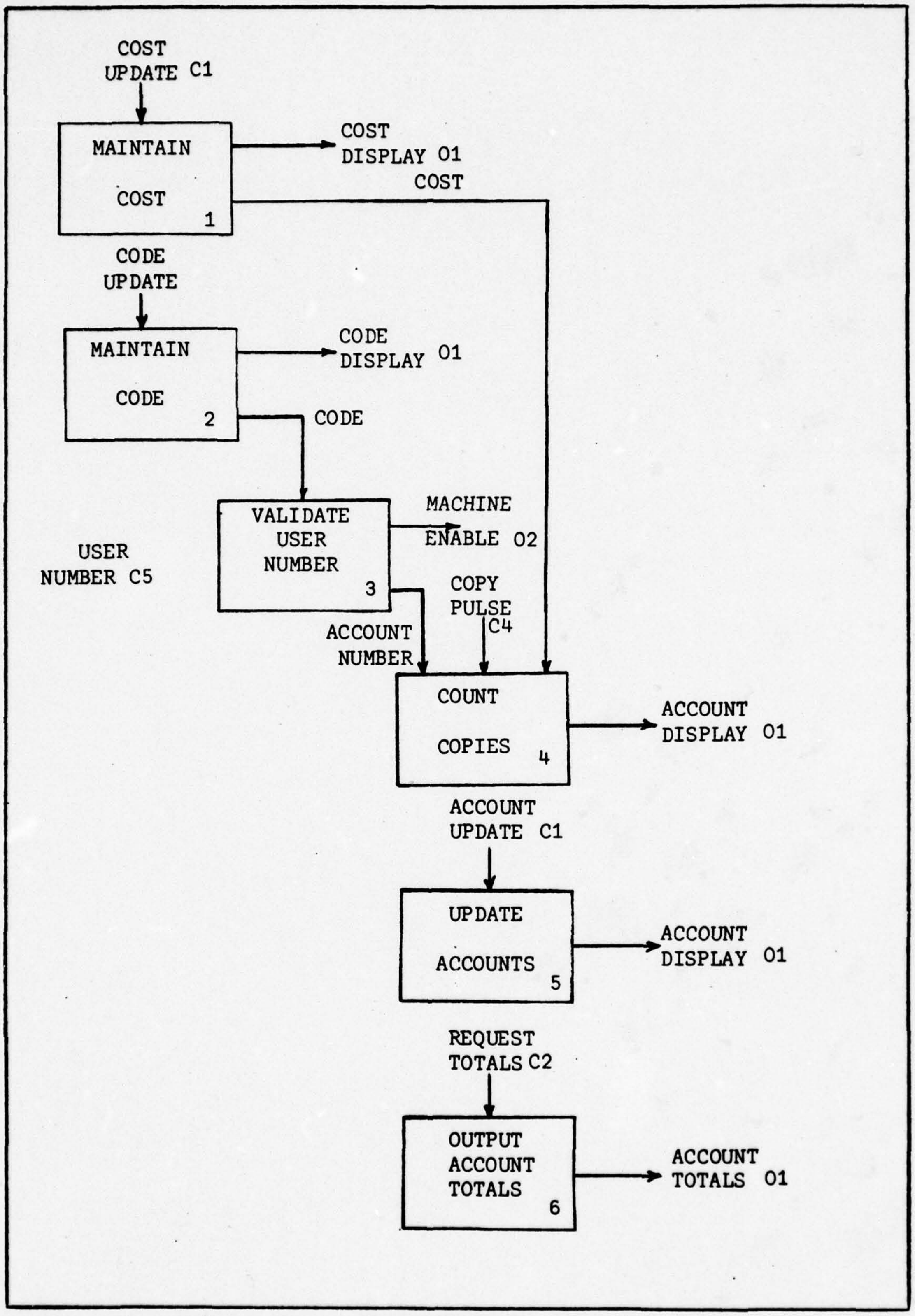


Figure II-4. Decomposed Xerox Controller Requirements

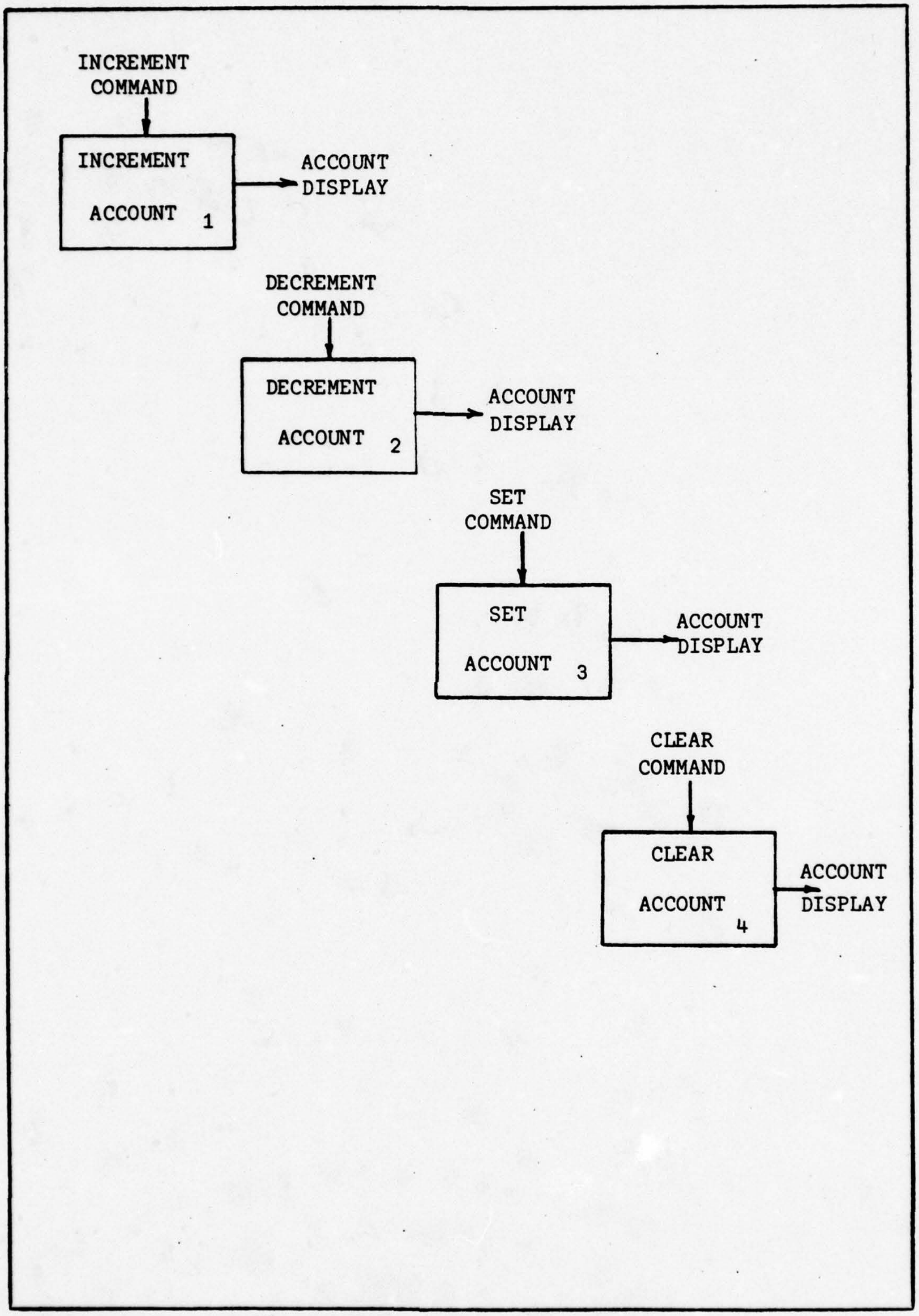


Figure II-5. Update Requirements

the controller must recognize. Boxes 1, 2, and 3 are self explanatory activities which operate on one account on each invocation. Box 4 (Clear Accounts) is required to initially set all accounts to zero so that the system can be put into operation. This function is also required if the treasurer chooses to zero the accounts after getting a listing of the totals.

The Display outputs which appear on several of the boxes are simply signals of completion of that particular requirement and their exact format will be determined in the design phase.

The exact requirements of the Xerox Controller are specified by the analysis performed in Figures II-1 through II-5. Though high level design decisions were made in this analysis, the attempt was made to avoid making low level design decisions prematurely. In addition to the functional requirements, some quantitative requirements were defined. The largest account balance experienced up to thebeginning of this project was $44.00 for a month. It was decided that account balances up to $99.00 would be allowed. A projected student population of just over 300 plus the small number of organizational accounts which exist led to the decision to require the system to handle up to 500 accounts.

Accounts Program Requirements

The Accounts Program requirements derived in Figure II-2 are shown separately in Figure II-6 for reference. The

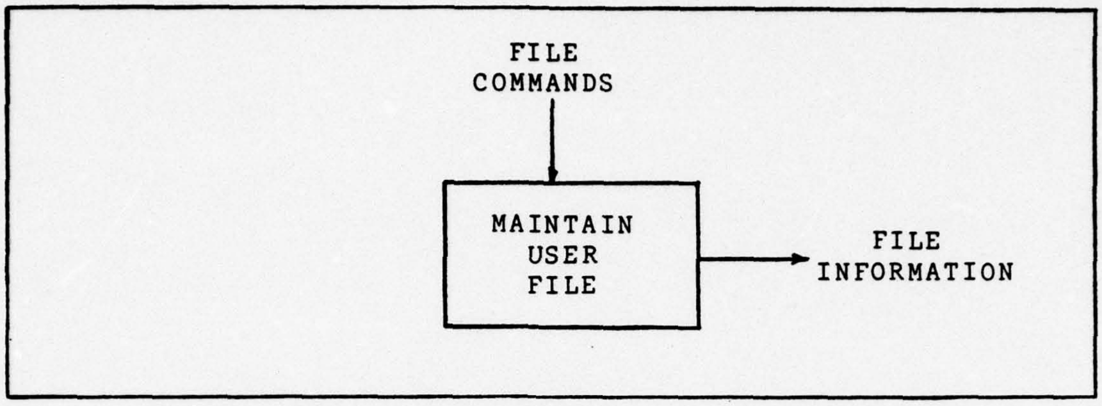


Figure II-6. Accounts Program Requirements

requirement to maintain user file is decomposed into its component requirements in Figure II-7.

The requirement to produce code implies producing a key to encrypt the account numbers, encrypting the account numbers to obtain user numbers, and producing a key which can be input to the Xerox Controller to facilitate decoding the user numbers. These requirements involved a design decision to use encrypted account numbers as user numbers. This decision is based on the premise that if the user number is not in some way decodable to produce the account number, then a list of valid user numbers must be maintained in the Xerox Controller. This would double the memory requirements in the controller which is an unreasonable price to pay for not having to encode and decode user numbers.

Boxes 2 and 3 of Figure II-7 are simply requirements to be able to add new accounts and change account information.

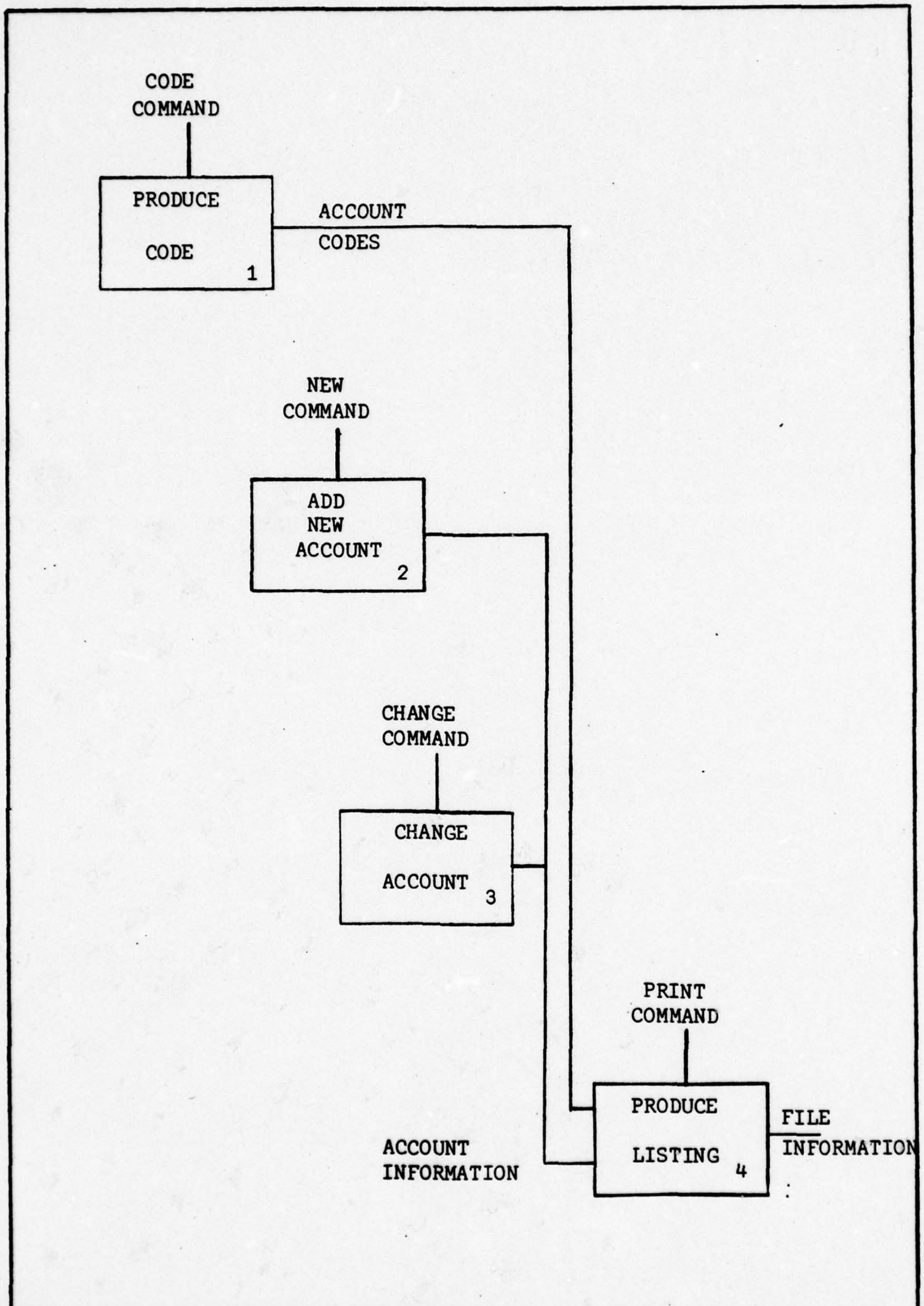


Figure II-7. Decomposed Accounts Program Requirements

The last box in Figure II-7 displays the requirement to take the account information out of the database along with the account codes and produce a listing of file information.

Quantitatively, the Accounts Program must be able to accomodate up to 500 accounts with 30 characters for identification and 10 characters for class designation. It must be an interactive program which leads the treasurer through its operation with only the minimum knowledge of the timesharing option on the CYBER system.

Summary

The system requirements have been defined and analyzed in this section using a graphical method similar to Softech's SADT method. By referring to Figures II-1 through II-7 and the quantitative requirements at the end of the Xerox Controller and Accounts Program sections, an understanding of the system requirements can be gained.

III SELECTION AND DESIGN OF XEROX CONTROLLER HARDWARE

Introduction

The Xerox Controller subsystem can be divided into functional modules which can then be implemented in software or hardware. The major portion of the hardware will be contained on a single board computer (SBC). The decision was made early in the project to use an SBC to eliminate the extensive trouble-shooting usually required when using a prototype board. The selection of the particular SBC will be directed at implementing as many of the functional modules as possible without extra circuitry. This chapter will cover the modularization of the Xerox Controller functions, the SBC selection, and the design of the external circuitry. For all diagrams in this chapter, a J indicates a connector on the SBC and the numbers in the J box are the pin numbers of the connector.

Xerox Controller Functional Modules

Figure II-4 graphically shows the requirements of the Xerox Controller. Each of the 6 boxes represents a functional module of the controller. In addition to these modules, the arrows entering and leaving the diagram define some hardware interface modules. All the update arrows, the User Number, and Request Totals arrows define an input device which may be part of the SBC. The last 3 arrows (hardware interfaces) are the Machine Enable, the Copy Pulse, and the Account Totals.

Abstractly, these will be implemented by a switch, a pulse detector, and a serial interface, respectively, and may possibly be part of the SBC. The Xerox Controller will be interrupt driven and to recognize only the desired interrupt (input device or pulse detector) will require an interrupt steering module. Figure III-1 shows graphically the functional modules described and is useful in gaining as understanding of the hardware described in this chapter.

Selection of Microprocessor

In this section an attempt is made to implement as many of the functional modules as possible on the SBC simply by judicious choice of a SBC. The first step in this selection process is the determination of the criteria to be used in making the selection. As in all designs, all other factors being equal, the lowest cost option will be selected. The criteria and the justifications are as follows:

1. Single power supply- The Xerox Controller will be self contained, cost and size must be held to a minimum.
2. Power switching capability- Enabling the Xerox machine requires switching a 110 volt power line.
3. Interrupt capability- The SBC must recognize the presence of an external signal to start processing.
4. Serial output capability- An RS 232C interface must be available to print out the account totals.
5. Display- Must be capable of displaying 4 or more digits in any readable format.

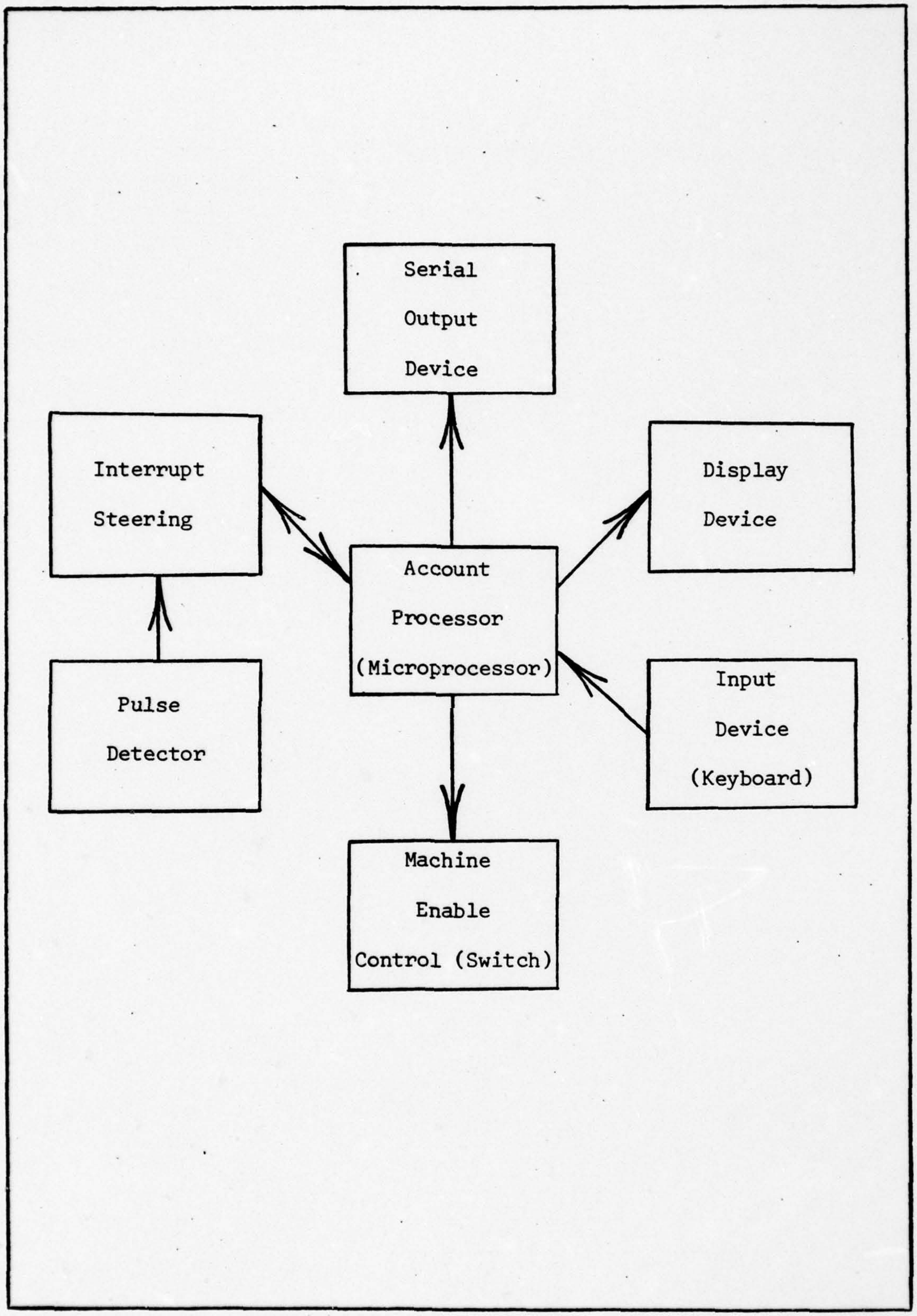


Figure III-1. Hardware Block Diagram

6. Development capability- A monitor program to handle entry and testing of Xerox Controller program.
7. Memory capacity- The controller must have a minimum of 1k bytes of RAM to handle the 500 account totals and 1k of EPROM to store the program.

The general trend in microprocessors has been towards more speed, more powerful instruction sets, reduced package count, and reduced cost. The demand for decreased package count led to the introduction of the 8048 family of computers on a chip by Intel (Ref 5). The 8035 is the basic version in the family and contains 64 bytes of RAM, an internal clock, latched ouput ports, an interval timer/counter, and all the external control lines to interface with almost any available device. Imsai used the 8035 to produce a control computer board which is tailored to a wide variety of control applications (Ref 3). The capabilities of the Imsai 8048CC relative to the criteria listed previously is shown below:

1. Single power supply- The 8048CC operates on a single 5 volt power supply.
2. Power switching capability- The 8048CC has 5 power relays capable of switching 4 amps at 220 volts.
3. Interrupt capability- The 8048CC has a single vectored interrupt. Though a multiple vectored interrupt system would be helpful, this could be simulated with software.

4. Serial ouput capability- The 8048CC has an RS 232C interface capability using the internal timer.
5. Display- The 8048CC has a 9 digit display and display controller which can be mounted off the board to satisfy the display requirement.
6. Development capability- The monitor program allows simple keyboard program entry and section by section debug execution mode.
7. Memory capability- The 8048CC has 2k of EPROM so that the 1k monitor program and the Xerox Controller program can both be maintained. The board has 1k of RAM with an additional 1k expansion capability.

The 8048CC meets all the criteria stated for selection of the microprocessor board. A number of other SBC's were considered but were rejected for not meeting one or more of the criteria. Three of the boards considered and reasons for rejecting them are presented below. These three are representative of the other boards considered but not presented here.

1. Intel 80/05- The 80/05 does not meet the power switching capability, the RAM capacity, or the display capability (Ref 1:118).
2. National Semiconductor SC/MP- The SC/MP does not have the power switching capability, the RAM capacity, or the display capability (Ref 1:130).
3. Technico SSS- The SSS is a 16 bit machine based on the TMS 9900 microprocessor. This board was rejected because it requires 3 power supplies (+5v, -5v, +12v).

Though this is a limited sample of the available SBC's, it shows the typical shortcomings of most of the computers. Lengthening the list to include those computers which were only remotely considered would not add any clarity to the selection process. The Imsai was designed as a control computer and is an excellent selection for the Xerox Controller.

Hardware Design

The following sections cover the design of the modules presented in the Xerox Controller Functional Modules section. Since this is the hardware design chapter only those modules which are implemented with hardware are presented. In addition to the hardware modules already mentioned, a memory power backup circuit to prevent loss of account information in the event of a power failure is described. A key switch circuit to allow only the treasurer to have access to the command mode of operation is also presented.

Memory Power Backup Circuit. The Imsai 8048CC contains 8 Intel 2102AL memory integrated circuits. The information to be stored in these devices is the code key, the copy cost, and the account totals. Information in random access memory (RAM) semiconductor devices is volatile, which means that if there is a power failure to the memory, the contents are destroyed. Since the account totals could represent a sizable monetary sum, a circuit is designed to prevent any power interruption from destroying the RAM contents.

Each of the RAM circuits requires 35mw of power at 2 volts

to maintain the security of the contents (Ref 4:2-34 - 37). The total power for all 8 of the memory circuits is then 280mw or 140ma at 2 volts. For a 24 hour period this equates to a power requirement of 3.36 amphours. Using the very conservative estimate of at most 7 days for a power interruption, the total power required is 24 amphours. Power Conversion produces a lithium power cell which was designed specifically for providing power backup for semiconductor memories. This power cell (model 660-5A Eternacell) provides 30 amphours at 2.8 volts and has a shelf life of 5 years minimum.

Figure III-2 shows the circuit which will be used to provide the power backup for the RAM. This circuit requires

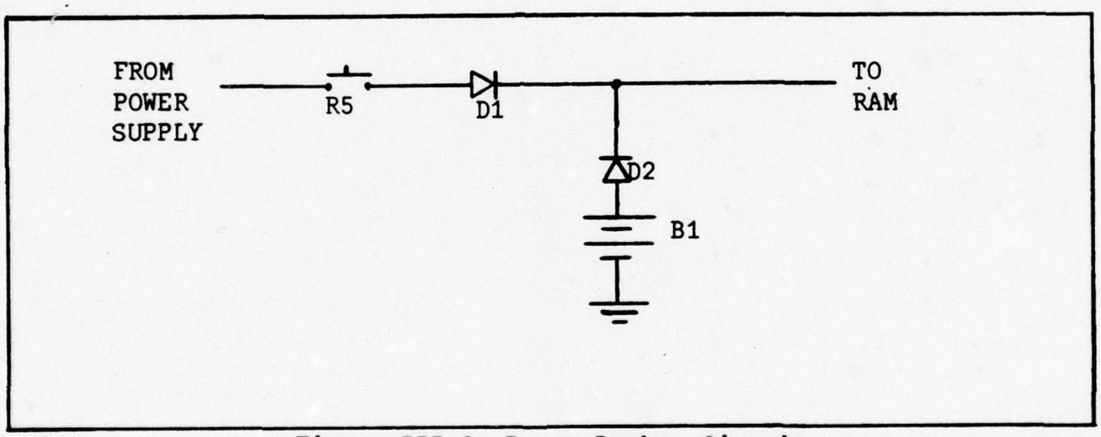


Figure III-2 Power Backup Circuit

breaking the power lead to the RAM section of the SBC and inserting the circuit. In normal operation 5 volts is supplied by the power supply through relay R5 which is powered closed whenever the power supply is operating. Diode D1 is forward biased and conducts power to the RAM. Diode D2 is reversed biased and no current flows through the battery. When a power interruption occurs relay R5 opens and disconnects the power

backup circuit from the remainder of the board. Diode D2 is now forward biased and conducts power from the battery to the RAM. Diode D1 is open but is needed during a power interruption to prevent the battery from supplying the power to keep the relay closed. When power is restored the relay provides a slight delay in reconnecting the power supply to the RAM so that power spikes from power up will not affect the memory contents.

This circuit is implemented by cutting the printed circuit which brings power to the RAM section of the board. A jumper is added which brings power to one of the normally open contacts of relay R5 from the power supply. The other normally open contact of the relay is connected to diode D1 and then to the power lead of the RAM section of the board. To facilitate battery replacement, diode D2 is soldered to the positive tab on the power cell. A 16 pin flat cable connector is attached to a short length (approximately 6 inches) of 16 conductor flat cable and the conductors to pins 9 and 10 are connected to the negative tab on the power cell and diode D2 respectively. The 16 pin flat cable connector is then simply plugged into any one of the sockets on the board intended for the expansion of the RAM being careful to match the pin numbers on the connector with those on the socket.

Machine Enable Control. Control of the operation of the Xerox machine will be maintained through the Polytech coin box mounted on top of the machine. This coin box has a single-pole-double-throw key switch which selects either coin operation or continous operation. The new system requires that the

switch be left in the coin operation position so that either a nickel or a user number will be required to operate the Xerox.

The 8048CC board contains 4 single-pole-double-throw relays which are connected to the output port 5. Figure III-3 shows a circuit which will allow output bit 2 of port 5 to take the place of the switch in the coin box. Since the coin box switch will be left in the coin operation position the wire to the upper position in the switch indicated in the figure with the X through it will be disconnected from the switch so that the relay will have full control of the switch operation. To

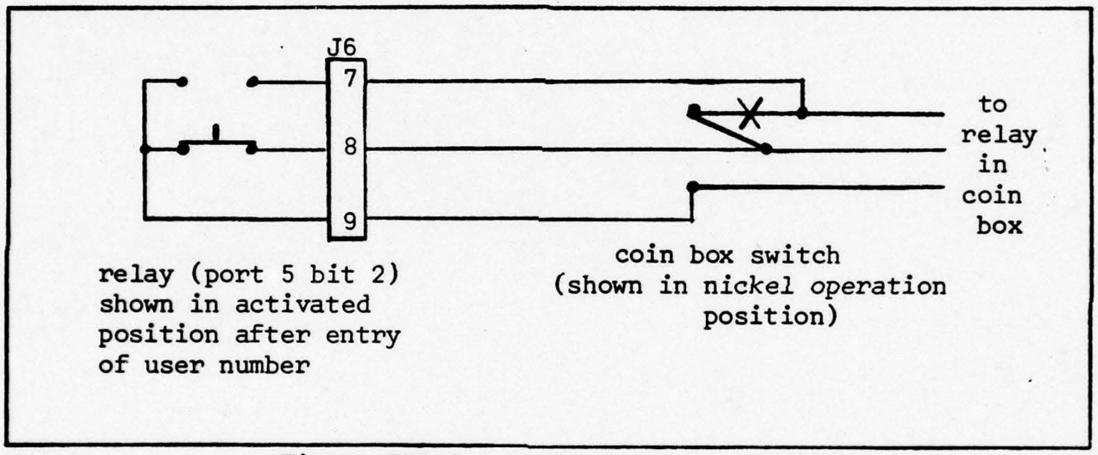


Figure III-3. Machine Enable Control

activate a relay requires outputting a zero to the appropriate bit of port 5. By arranging the contacts in the relay so that continous operation is commanded by the activated position, a power failure to the Xerox Controller will put the Xerox machine in coin operation. If the Xerox Controller is disconnected from the coin box the wire which was disconnected will have to be reconnected or the machine will only operate with the key switch in the continous operation position.

Copy Pulse Buffer Circuit. Circuit diagrams were not available for the Xerox machine or the coin box to help determine the proper location to wire into to detect when a copy has been made. A signal was found which is apparently used to trigger the solenoid on the mechanical counter in the coin box. Figure III-4 is a sketch of the waveform detected in the coin box every time a copy was made by the Xerox machine.

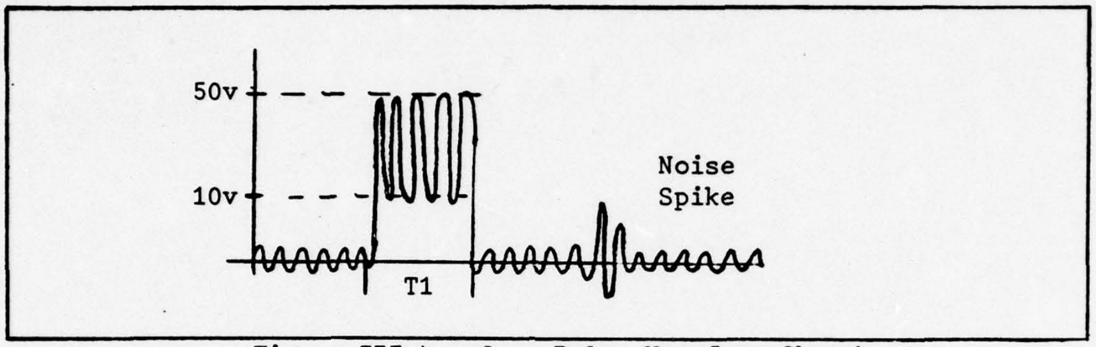


Figure III-4. Copy Pulse Waveform Sketch

The waveform is apparently produced by switching an alternating current 60 Hz signal on for the time T1 when a copy is made. This signal is rectified and filtered to give the signal portion shown for time T1 which has a 30 volt direct current portion to trigger the solenoid. The time T1 was found to vary between approximately 125 milliseconds and 230 milliseconds. The noise spike depicted in the sketch is typical of the waveform resulting from the on and off operation of unknown devices within the Xerox machine. The waveform necessary to cause an interrupt to be recognized by the computer is shown in Figure III-5. The time T2 must be long enough for the computer to detect the interrupt but not so long that it is present when the interrupt routine has been completed and the interrupt system is reenabled. The 8048CC computer has an instruction cycle

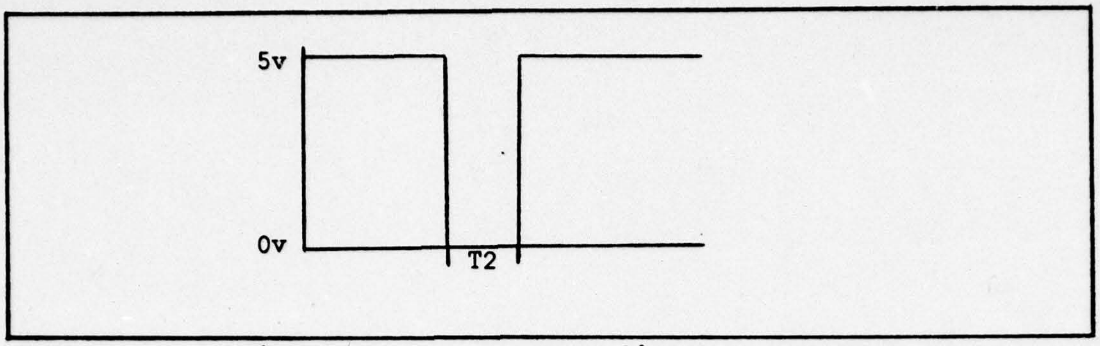


Figure III-5. Interrupt Signal to Computer

time of 4.1905 microseconds (Ref 3:) with some of the instructions taking 2 cycles to complete. Since 2 cycle times could pass before an interrupt signal is detected, the signal must persist for at least 2 cycle times. The Count routine which is executed in response to an interrupt caused by a copy pulse is 147 instruction cycles long, so T2 must in no case be longer than 147 cycle times. The allowable range of T2 times is then approximately 8.5 microseconds to 616 microseconds.

The circuit in Figure III-6 is the pulse shaping circuit which is to produce a signal capable of triggering an SN74121 integrated circuit (IC) (Ref 11:134-137). The resistor

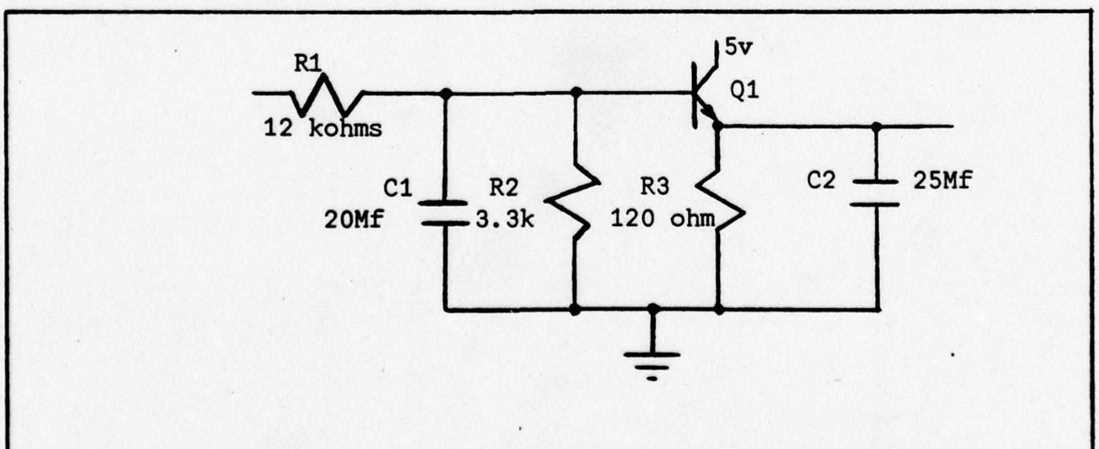


Figure III-6. Pulse Shaping Circuit

divider formed by resistors R1, R2, and R3 has the effect of attenuating the 30 volt dc portion of the signal down to 3.1 volts. The transistor Q1 and Resistor R3 form an emitter-follower circuit which provides the triggering signal to the input of the SN74121. Capacitors C1 and C2 are in the circuit to attenuate the 60 Hz portion of the signal and the noise spikes so that the 1.55 volt threshold for the SN74121 is not crossed causing a false interrupt.

The remainder of the Copy Pulse Buffer Circuit consists of 2 of the SN74121 monostable multivibrators (one shots) connected as shown in Figure III-7. The one shots produce a high going pulse on the Q output and a low going pulse on the $\overline{Q}$ output when the B input passes through 1.55 volts rising. The length of the pulse produced is determined by the values of the resistors and capacitors shown in the figure. The circuit U2 is used to produce the proper length pulse to input to the interrupt line of the computer. Initially U1 is at rest and as such the Q output is at a logical 0 level. This makes both a inputs to U2 logical 0 and enables an input on the B input to trigger the one shot. The Q output from U2 is fed back into the B input of U1 which is always enabled since the A inputs are tied to ground. When U1 triggers in response to the rising B input its Q output goes to a logical 1 level causing the A inputs of U2 to go to a logical 1 level and disable the B input. The shortest elapsed time between copy pulses received from the Xerox machine was measured to be slightly in excess of 1.2 seconds. The length of time which

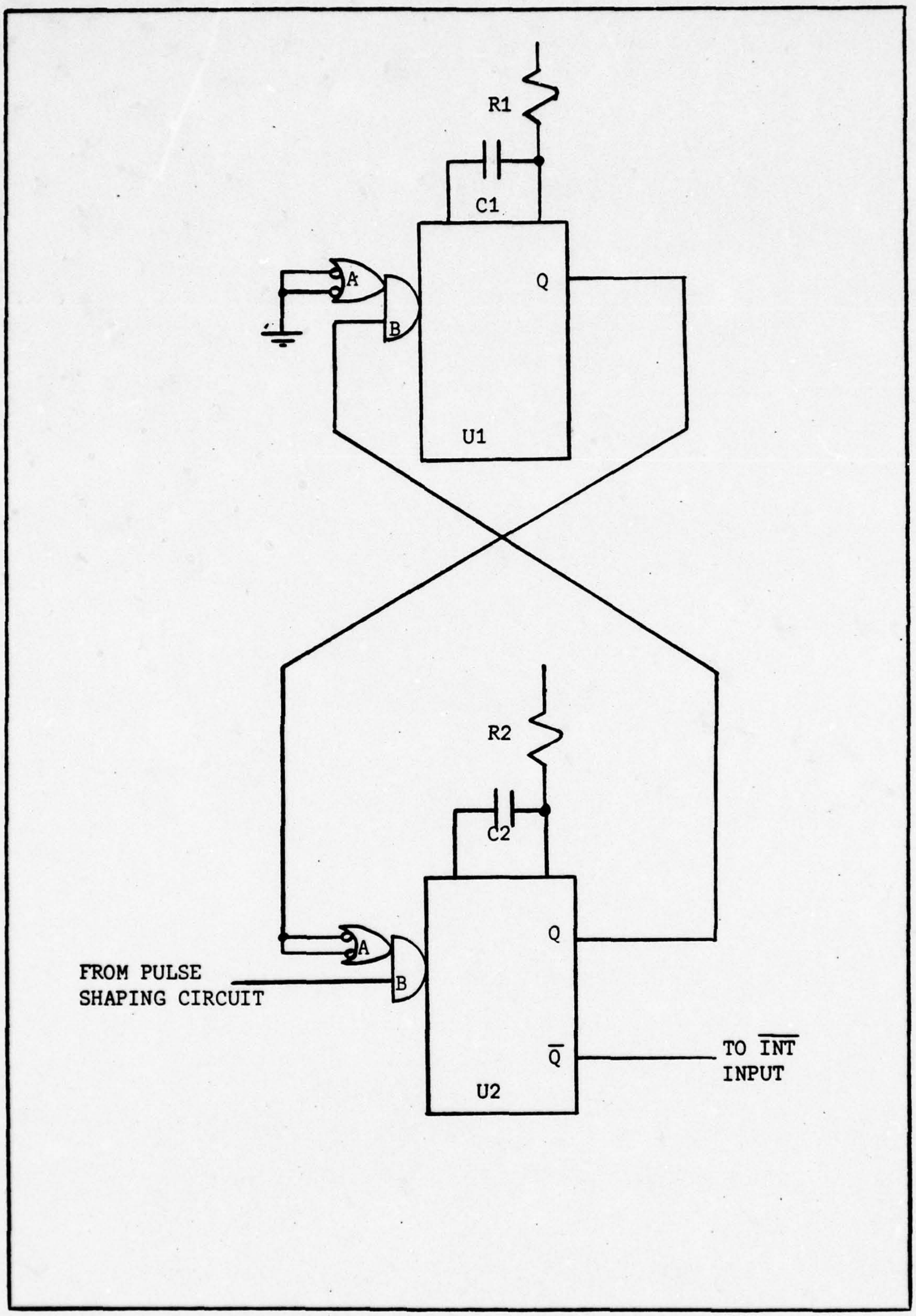


Figure III-7. Interrupt Pulse Circuit

U2 must be disabled by U1 is greater than T1 from Figure III-4 and less than 1.2 seconds. To provide a reasonable amount of noise immunity the pulse duration selected is 800 milliseconds. The pulse duration determined by the resistor and capacitor network attached to the one shots follows the equation

$$t(out)=CR \ln 2 \tag{1}$$

Using a 33kohms resistor and a 35 microfarad capacitor produces the desired 800 millisecond pulse. The required pulse duration for U2 has already been specified to be between 10 and 600 microseconds. With a resistor value of 8200 ohms and a capacitor value of .01 microfarads equation (1) yields a pulse duration of 57 microseconds.

The Copy Pulse Buffer Circuit is mounted on a seperate small board attached to the computer board on offsets. The copy pulse signal line and ground are brought in on connector J6 pins 24 and 26 and jumpered to the small board.

Interrupt Steering Circuit. The 8048CC must accept interrupts from either the keyboard or the Copy Pulse Buffer Circuit but not both. One of the single-pole-double-throw relays attached to port 5 is used as a software controllable switch. Figure III-8 shows the circuit necessary to implement this steering function. The 8048CC User Guide (Ref 3) section on jumpers was followed regarding allowing the interrupt from the keyboard controller to drive the interrupt input except that the relay is placed where the jumper is supposed to be.

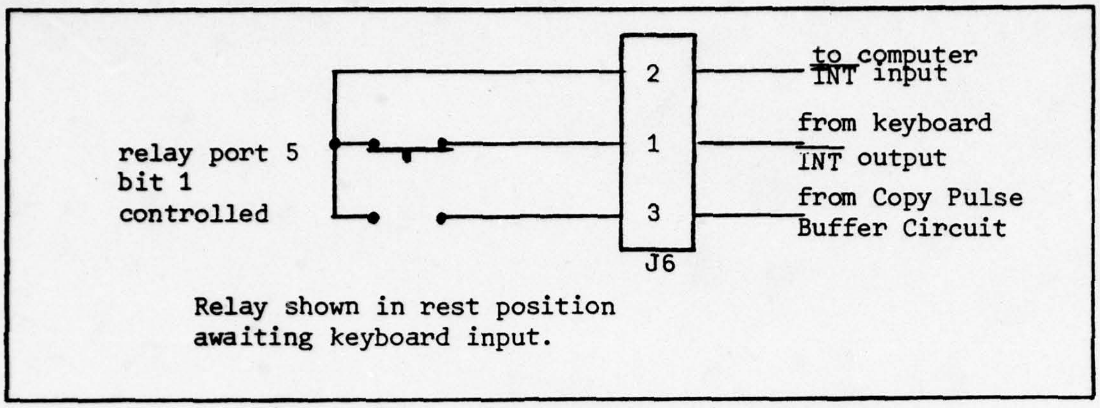
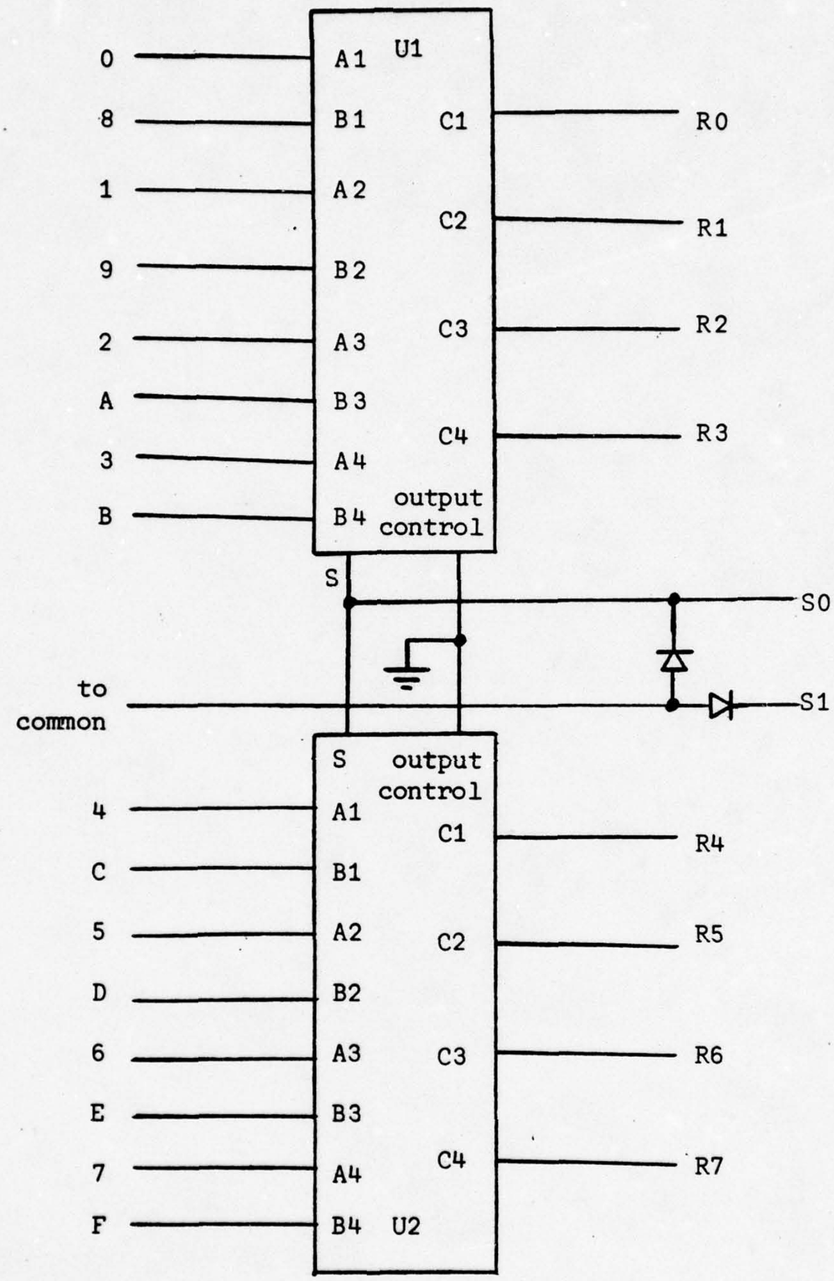


Figure III-8. Interrupt Steering Circuit

<u>Keyboard Circuit</u>. The design decision was made to use an available hexadecimal keyboard rather than the keyboard on the 8048CC board because the keyboard on the computer board does not provide any mechanical feedback indicating that an input has been accepted. The keyboard controller (Intel 8279) and associated circuitry on the computer board is designed to read a keyboard which is arranged in a matrix with 3 rows of 8 switches. The external keyboard has only the hexadecimal digits 0 through F which are in the first 2 rows of the matrix. It also has only 1 common terminal instead of 2 (1 for each row). Figure III-9 shows a circuit using two SN74LS257 data selectors which simulates a scanned matrix keyboard. A ground on S0 or S1 indicates that row is being scanned while a ground on one of the R lines indicates that that column button of the scanned row has been depressed. From the circuit diagram it is apparent that when either S0 or S1 is low, the common terminal on the keyboard is pulled low. If one of the A lines into the data selectors is low when S0 is low, a low will be placed on the appropriate R line. The same effect occurs with the B lines and S1.



U1 and U2 are SN 74LS257 Data Selectors

Figure III-9. Keyboard Circuit

User/Control Switch Circuit. A key switch is mounted on the Xerox Controller which the treasurer can use to place the computer in the control mode. The control mode is used to change any of the system parameters and it must not be possible for anyone without the key to get access to this mode of operation. The computer has a testable input line to which this switch is connected. A 1000 ohm resistor is placed between the line (T1) and the 5 volt power supply to maintain a high level on the line when the switch is in the open user mode position as it is in Figure III-10. It should be noted that if a break occurs in the off board wiring, the line will remain high and consequently in the protected user mode.

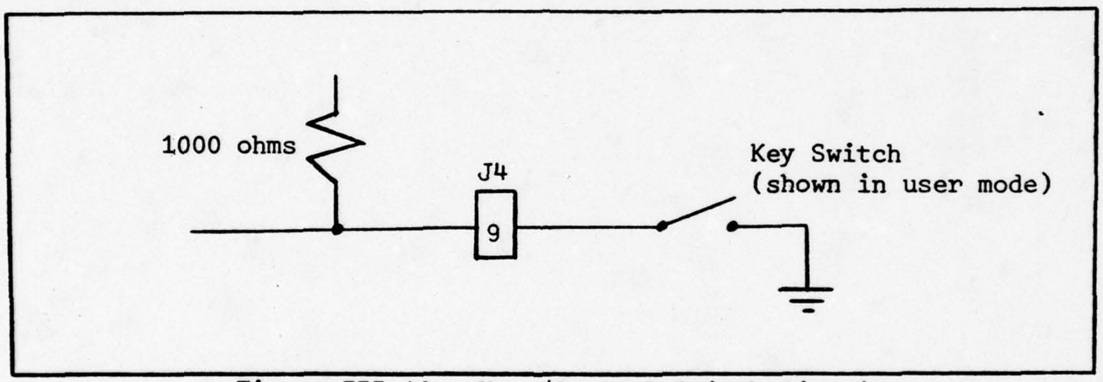


Figure III-10. User/Control Switch Circuit

IV DESIGN OF XEROX CONTROLLER SOFTWARE

Introduction

The Xerox Controller program was designed using the Structured Design (SD) technique popularized by Yourdan and Constantine (Ref 12) because its use leads to functionally well defined and easily modifiable program units. No cross-assembler could be located, so assembly was accomplished manually using the text substitution capability of the CYBER system timesharing utility.

The design effort was divided into a module specification phase and a module design phase. This chapter is consequently divided into two sections with the module design phase further divided into sections for each module.

Module Specification

The overall requirements for the Xerox Controller are presented in Figure II-4. Using that as a guide, the bubble chart representation in Figure IV-1 was created to subdivide the system into easily understandable modules. The brace which goes through the Keyboard Inputs arrow indicates an afferent branch. This branch in the structure chart has keyboard inputs as its data item passing up from a module which specifies a keyboard read. Conversely, the braces which go through the User Number and Treasurer Commands arrows specify efferent branches. These two branches from the Executive module have as data items the User Number and Treasurer

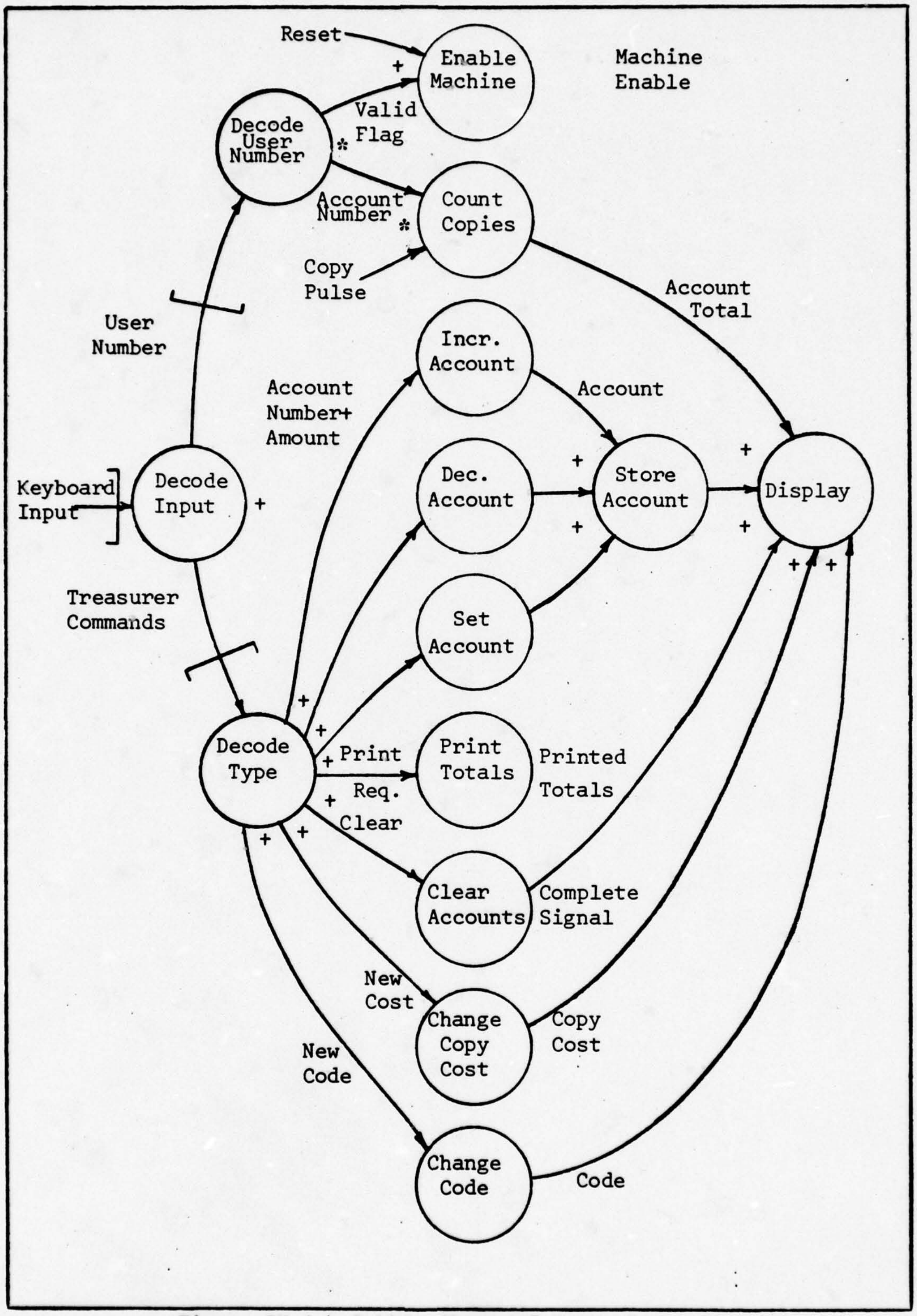


Figure IV-1. Bubble Chart of Xerox Controller

Commands. Since the bubble "Decode Input" is between the braces, it should be a central transform module; however, it involves only testing the User/Control switch position and it is absorbed into the Executive.

The Decode Type bubble is a transaction center. The OR symbols between the arrows exiting the bubble indicate that, after the command type is decoded, the appropriate arrow is followed. If the command is an Increment, Decrement, or Set Account Command, the output arrow is an Account and is input to the Store Account bubble, which is a shared module. The Account Total, Copy Cost, Code, or Complete Signal is input to the shared module, Display, to be displayed on the 7 segment display.

Decode User Number is the first bubble in the other efferent branch. The input is the User Number and the two outputs are the Account Number and the Valid Flag. The symbol between the two outputs is the AND symbol and indicates that both outputs are produced by the module. The Count Copies bubble must take the Account Number AND the Copy Pulse and increment the account by the copy cost and send the new Account Total to the shared Display module. The Enable Machine module takes either the Valid Flag or the Reset signal and enables or disables the Xerox machine respectively.

The Structure Chart of Figure IV-2 is derived from the bubble chart and every module is taken directly from the chart except the function Fetch Account which is factored out of the Increment, Decrement, and Set Account and Print Totals modules.

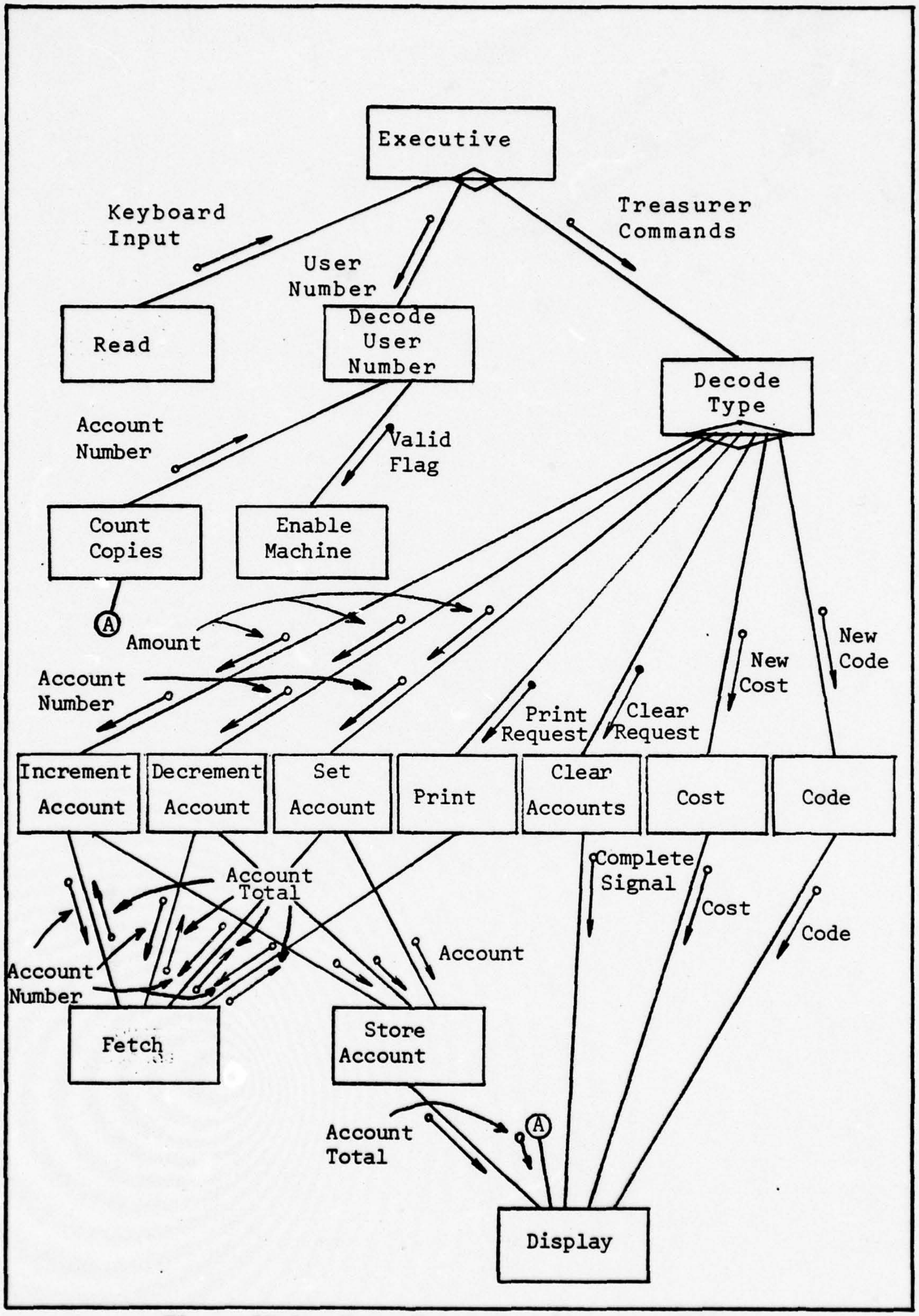


Figure IV-2. Xerox Controller Structure Chart

The modules of the Xerox Controller program are completely specified by the bubble and structure charts of Figures IV-1 and 2. The following section is devoted to the design of each module.

Module Design

The combined source and object listing of the Xerox Controller program is contained in Appendix A. A complete description of the assembly language and the machine language of the 8048 family of microprocessors is contained in the Intel MCS-48 Microcomputer User's Manual (Ref 5). Most of the modules are simple in design and they will be simply explained without resorting to flow charts or other design aids. When a section of code is referred to by a label it applies to the code from that label to the next label.

Executive. The Executive module consists of the code labeled RESET, INT, RST, STBY, and USR. RESET (location 000) is where execution starts after receipt of a reset pulse. From RESET, execution continues at RST where the relays attached to port 5 are set so the Xerox machine is not enabled (bit 2) and the Interrupt Steering circuit (bit 1) is set to receive keyboard entry interrupts. The remainder of the RST section sends commands to the 8279 Keyboard/Display Interface (Ref 5: 7-117) to prepare it for operation, initializes the timer flag, and sets the R0 register to access the Keyboard Interface. STBY is simply a loop which enables the interrupt system and then waits for an interrupt.

When an interrupt is received, the INT and USR section of the Executive are entered. The first instruction of the INT section of code determines whether the Xerox Controller is in the User or Control mode and jumps to USR or continues in the INT section as appropriate. Both of these sections of code involve successive calls to some entry point in the Read module which reads the next keyboard input and places it in the correct register for later use.

Read. The Read module is a state dependent module. This means that the module has multiple entry points and where it enters on a call is determined by its previous invocation. The Read module incorporates the code labeled FIRST through 8th, USR1 through USR6, ECHO, and READ. ECHO and READ simply take a keyboard entry, return it in the accumulator, and send a hexadecimal digit to the display. ECHO is used in the Control mode and the input digit is echoed on the display. READ is used in the User mode and for each input it returns the digit in the accumulator and sends 8. to the display.

The accumulator is used as the state pointer to control the state of the Read module. Initially the accumulator is set to zero in RST so when an interrupt is received, depending on the position of the User/Control switch, the entry point into the Read module is USR1 or FIRST. Before returning from the Read module the accumulator is set so that the next interrupt will cause entry into the Read module at the next state entry point (USR2 or SECOND). At the end of USR6, execution is sent to the Decode User Number module and after 8th, the next

keyboard interrupt causes entry into the Decode Type module.

If a User Number has been entered, the 6 digits end up in registers R2, R3, and R4. If a Command has been entered the first digit, which is the type code, is in register R7. The next 3 digits, which is the Account Number in any of the account accessing commands, ends up with the first digit in R2 and the second and third digits in register R3. The last 4 digits of the command are placed in registers R4 and R5.

Decode User Number. Because of the sensitive nature of this module, which decrypts the User Number, it will be presented in Appendix Z which will have restricted access and can only be obtained from the author.

Decode Type. The Decode Type module takes the Type Code in register R7 and sends execution to the appropriate module for processing. The codes 1 through 7 refer to the commands Increment Account, Decrement Account, Set Account, Print Totals, Clear Accounts, Set Code, and Set Cost respectively. The section of code which accomplishes this task is contained in the sections labeled TEST through T7 and its operation is self explanatory.

Increment Account. The Increment Account module increments the account specified in registers R2 and R3 by the amount contained in registers R4 and R5. The first step in this process is to call the Fetch module which, returns the account total in R6 and the accumulator and the account address in R2 and R0. A simple double byte binary coded

decimal (bcd) add is accomplished and if there is not an overflow (carry), the result is passed to the Store module to replace the current account total with the new one. The code for this module is contained in the section of code labeled INC.

Decrement Account. The Decrement Account module operates exactly as the Increment Account module does except that it does a bcd subtract instead of an add. The 8048 instruction set does not include a subtract instruction so one had to be fabricated. A bcd subtract was produced by taking the nines complement of the account total, adding the amount to be decremented, decimal adjusting the result and again taking the nines complement. Taking a nines complement of a hexadecimal digit is accomplished by adding 6 to each digit and complementing. For example, the nines complement of 6 is obtained by adding 6 for a hexadecimal total of C and taking the complement for a result of 3. If an underflow is indicated by a carry in the add operation the decrement is aborted and the result is not stored or displayed. The code for this module is contained in the section labeled DEC.

Set Account. The Set Account module is simply a call to the Fetch module to obtain the account address and a jump to the Store module to place the desired total in the account and display it. The code for this module is contained in the section labeled SET.

Print Totals. The Print Totals module is rather complicated and requires more than an explanation to present it. Figure IV-3 is a flow chart of the overall control flow. The routines Heading and Trailer print strings of ASCII characters before printing account totals and after completing the printing of the totals. The Fetch module retrieves an account and the PRINAC module formats and prints the account number and total.

Figure IV-4 hsows the Heading and Trailer control flow. The messages stored at Heading Address and Trailer Address are strings of ASCII characters which form the heading and completion messages respectively. The Printer module is a module which prints out the particular message whose address is in the accumulator.

Figure IV-5 shows the PRINAC module (print account) control flow. The boxes which Call ASCII for the multiple digits are actually a call for each digit but for simplicity are shown as single boxes. The Printer call is as previously described and the SEROUT module is used to output the ASCII code for the decimal point.

Figure IV-6 shows the control flow for the Printer module. It simply outputs successive ASCII characters of the respective message using the SEROUT module until it detects the BEL code (07) which is used to mark the end of message.

Figure IV-7 is the ASCII module which converts hexadecimal digits to ASCII code for printing. This was accomplished by adding 30 to the digit and decimal adjusting the result. The decimal digits then became 30 through 39 and the hexadecimal

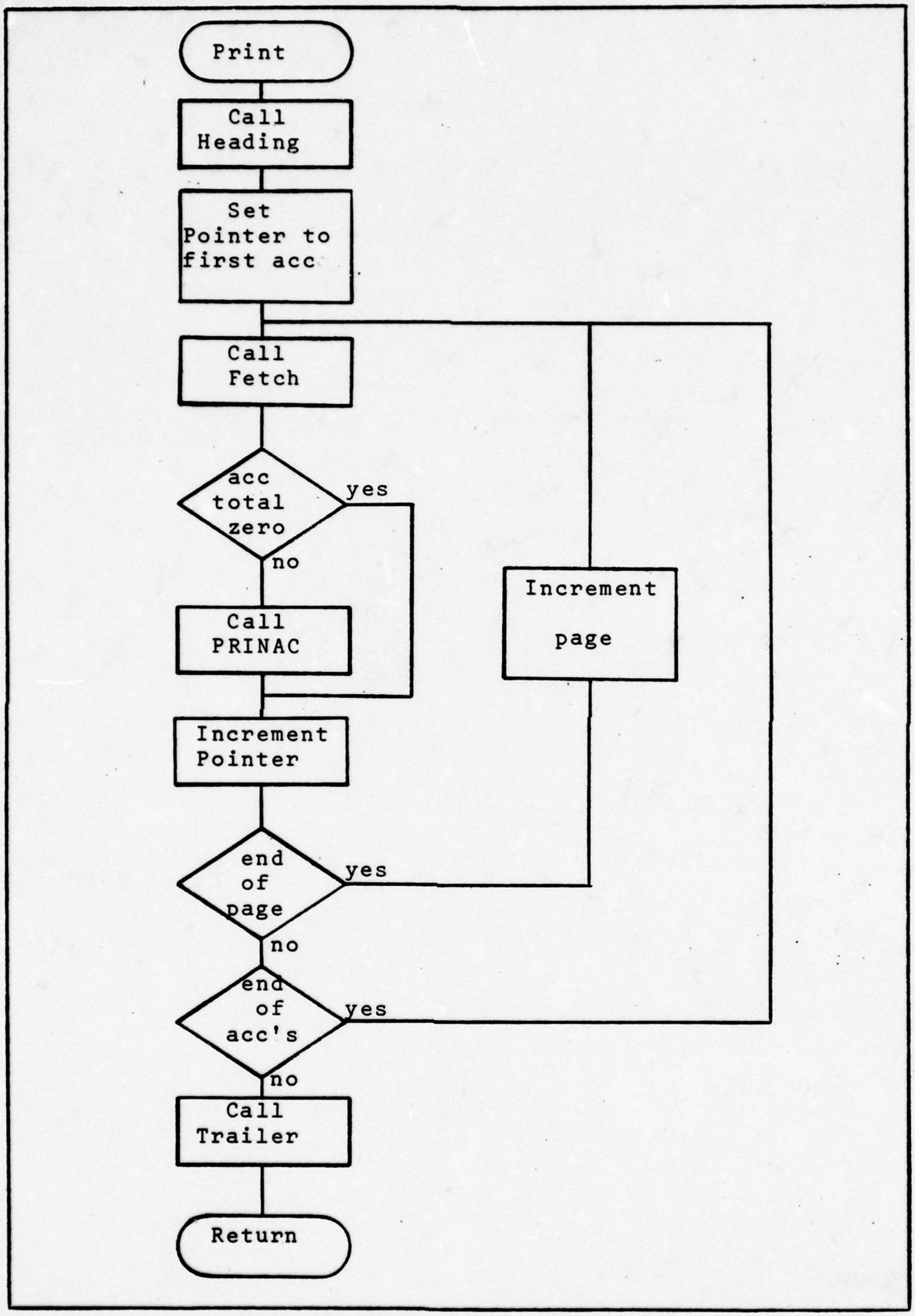


Figure IV-3. Print Flow Chart

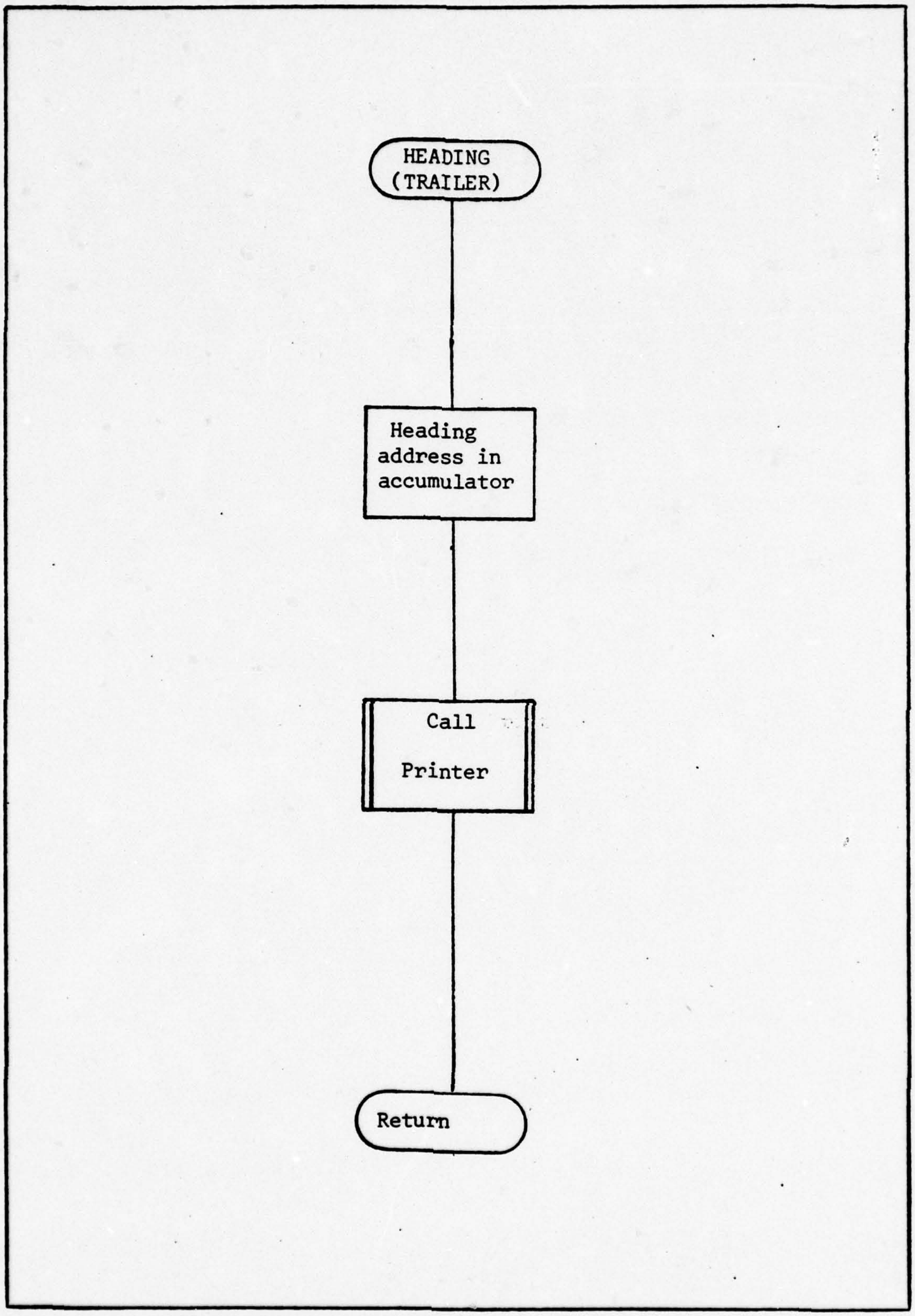


Figure IV-4. Heading (Trailer) Flow Chart

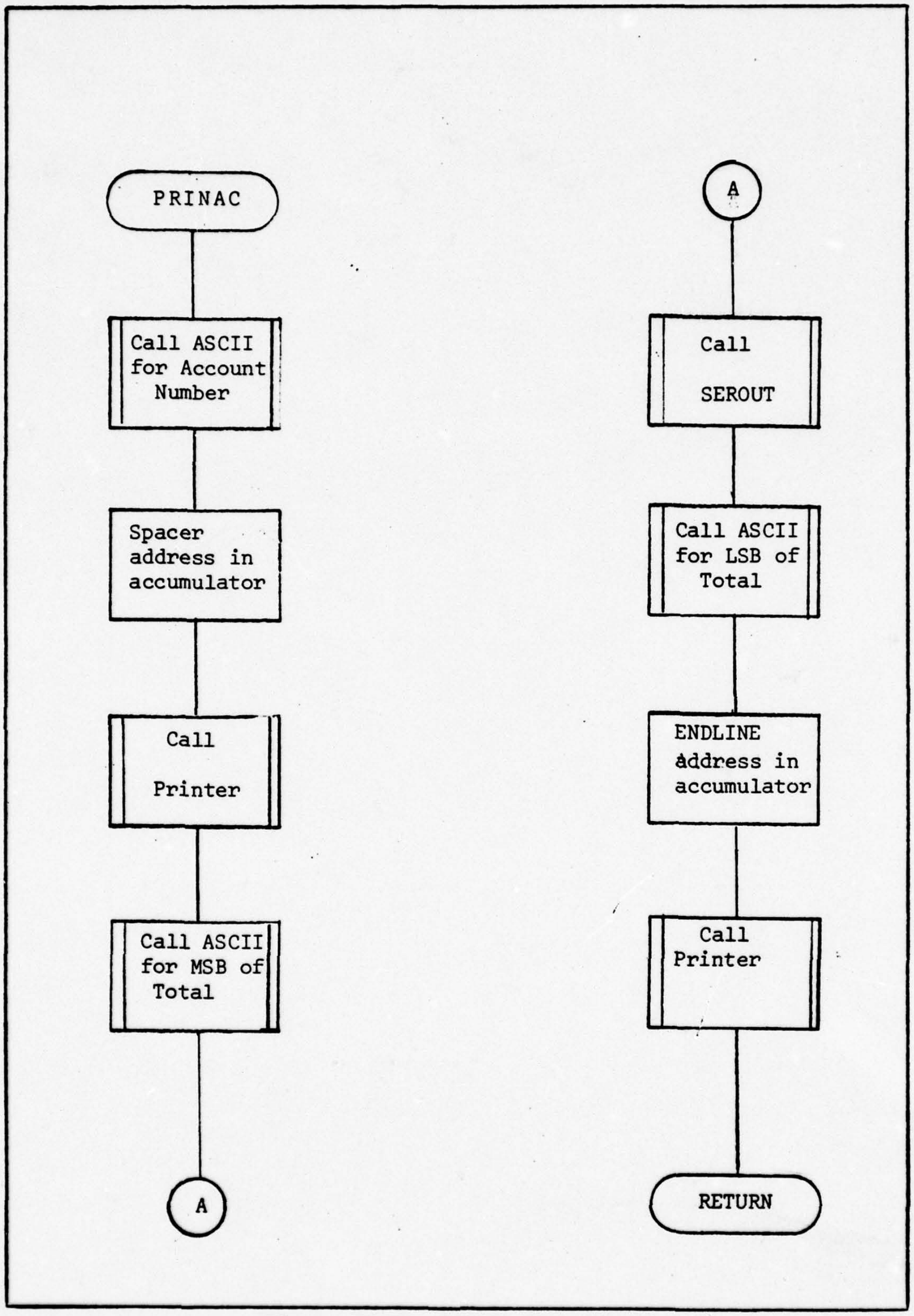


Figure IV-5. PRINAC Flow Chart

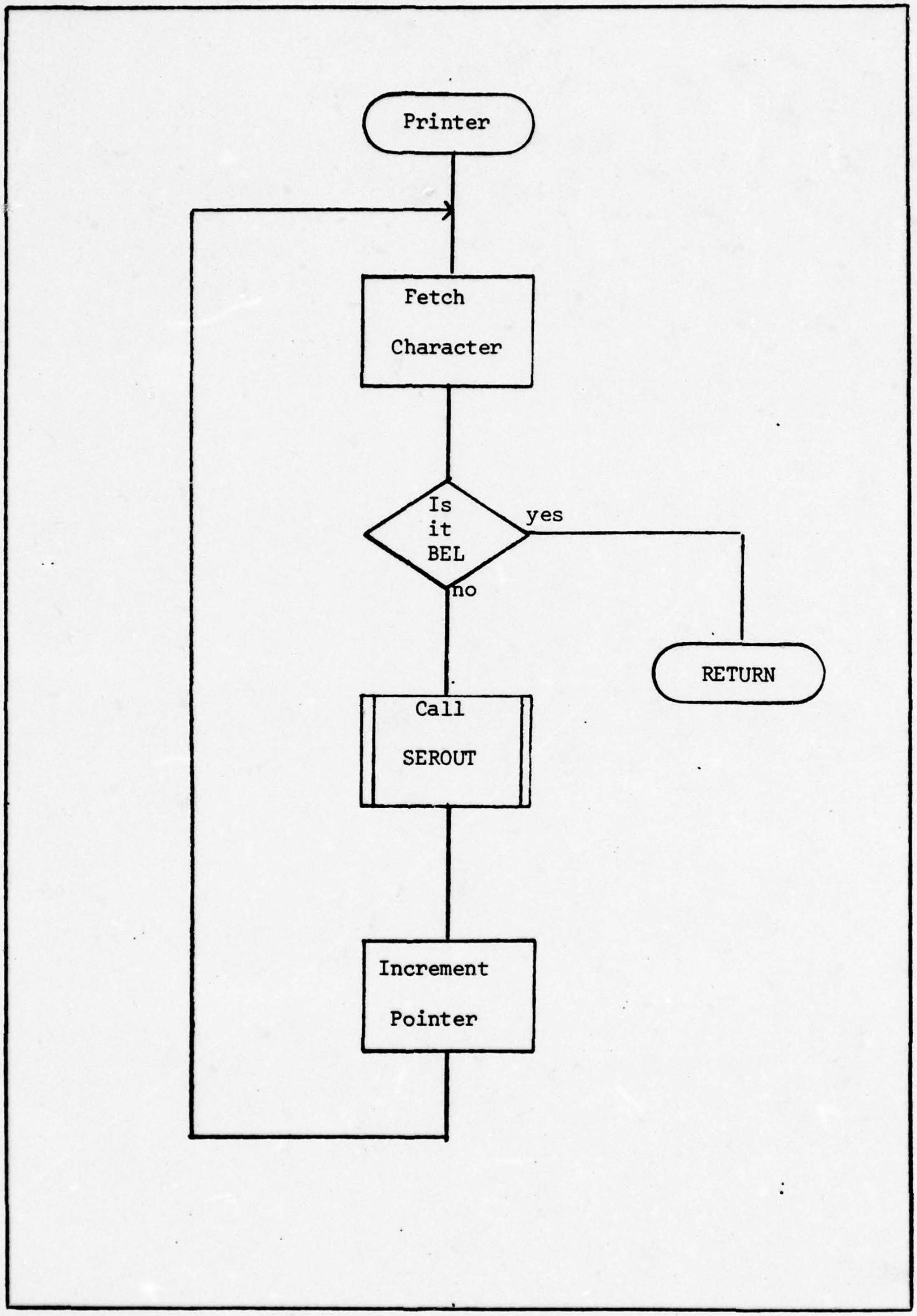


Figure IV-6. Printer Flow Chart

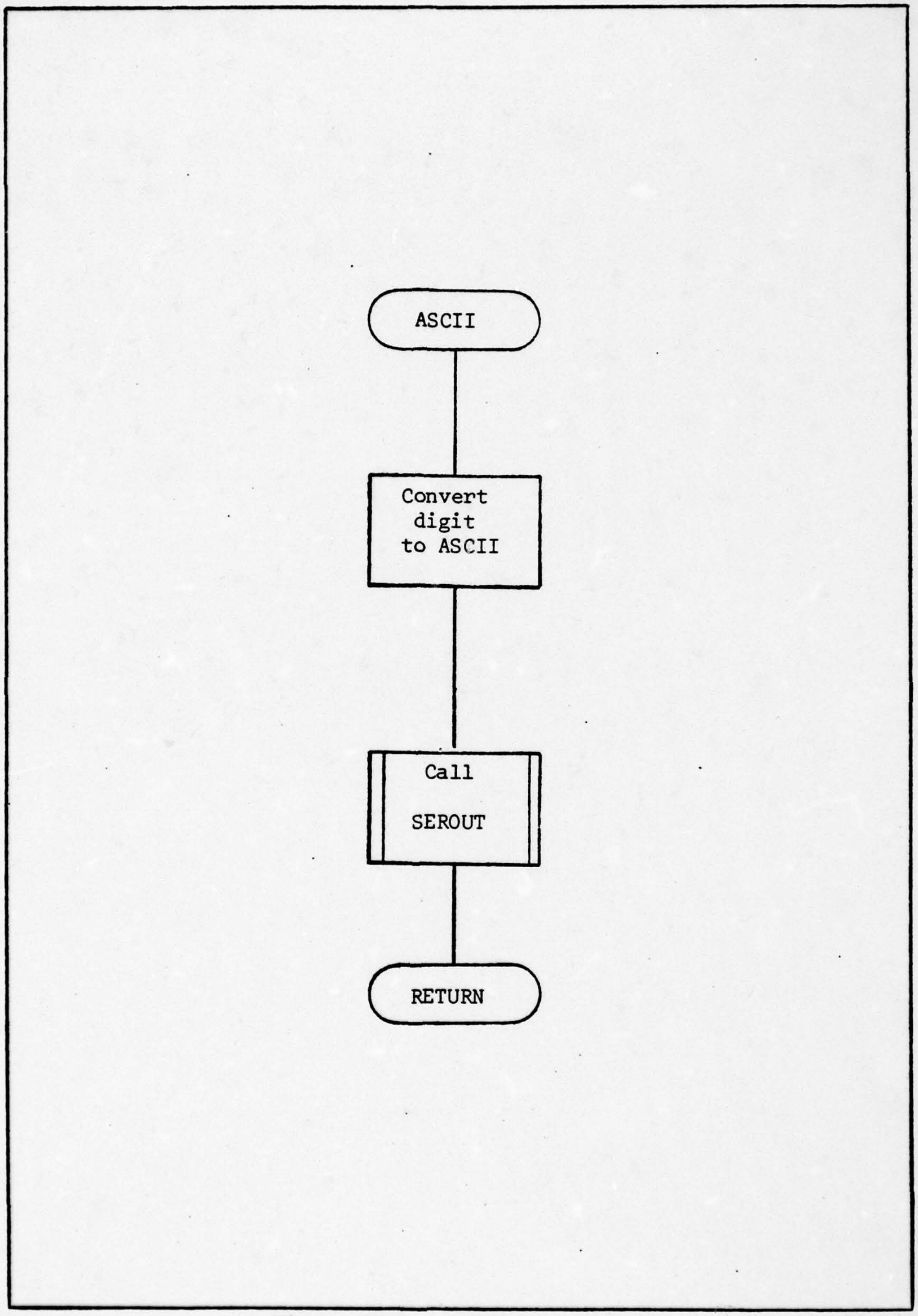


Figure IV-7. ASCII Flow Chart

digits A through F become 40 through 45. Any result which has a 4 as the first digit is then incremented by 1 to correct it to the proper 41 through 46 for A through F.

The SEROUT module is shown in Figure IV-8. The bit pointer is set to 11 (0B hexadecimal) so that the output of a character will consist of a start bit, the 8 bit ASCII code, and 2 stop bits. The RS 232C serial output is from port 7 bit 2 so that bit will be manipulated to produce the desired output. It should be noted that for the serial output a 0 level is output for a 1 in a bit position and the start bit is a 0 and the stop bits (2) are 1's. The DELAY the TIME module is called with will be determined in the discussion of the TIME module.

The actual operation of the TIME module is self explanatory, but the determination of the delay requires some explanation. The primary output device will be the TI silent 700 terminal (model 735) and it will operate at 300 baud. The basic instruction cycle time of the 8048CC is 4.1905 microseconds and the timer rate is 1 count every 134.095 microseconds. The number of instruction cyle times between when the timer flag is set and when the timer is next started must be counted using the table giving the number of cycles per instruction (Ref 5:6-5). There are 19 cycles giving a time of 79.6 microseconds. The time between bit changes at 300 baud is 3333.3 microseconds leaving 3253.7 microseconds to be accounted for by the TIME module. Using 24 timer counts (18 hexadecimal) gives a delay of 3218.3 microseconds.

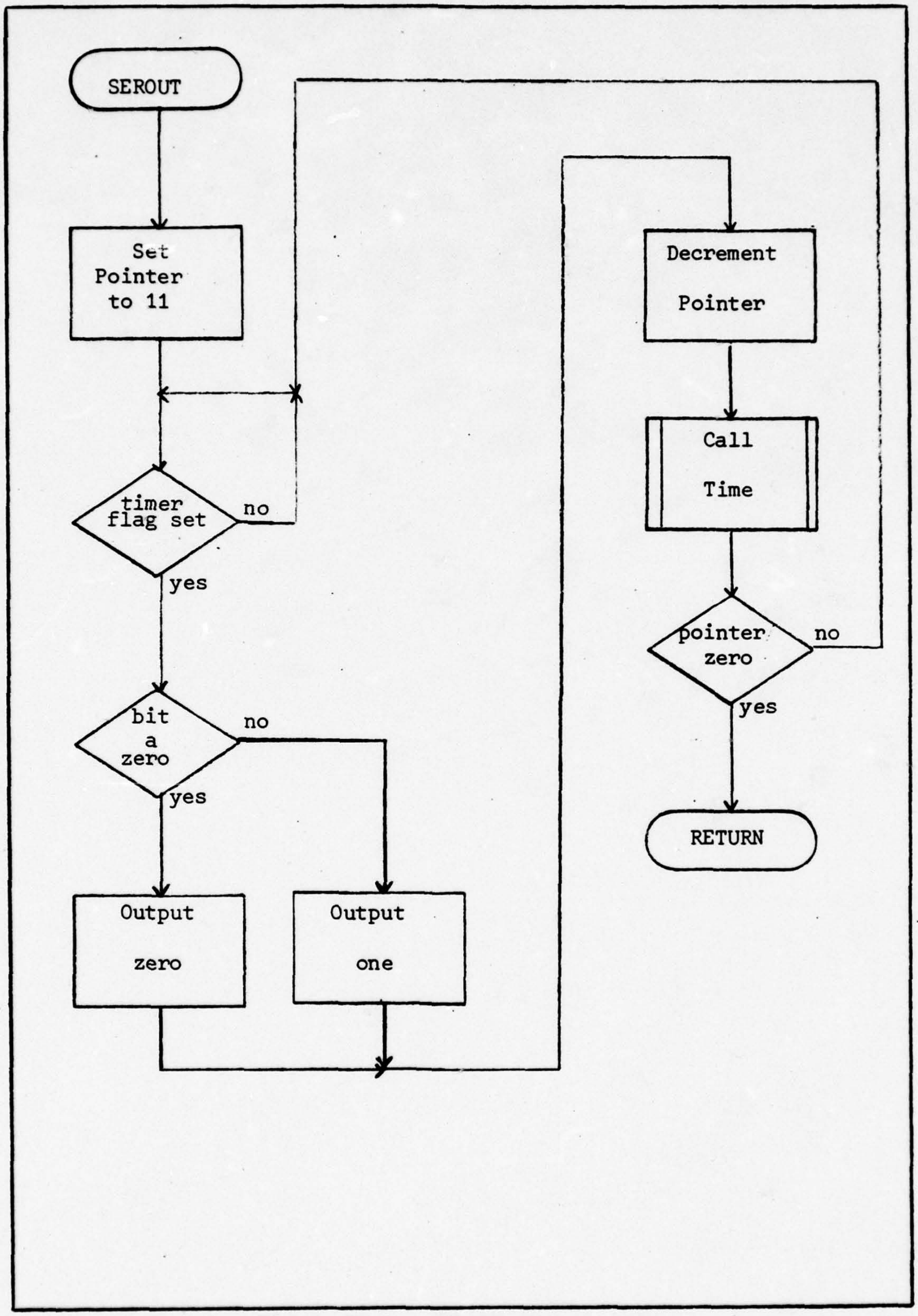


Figure IV-8. SEROUT Flow Chart

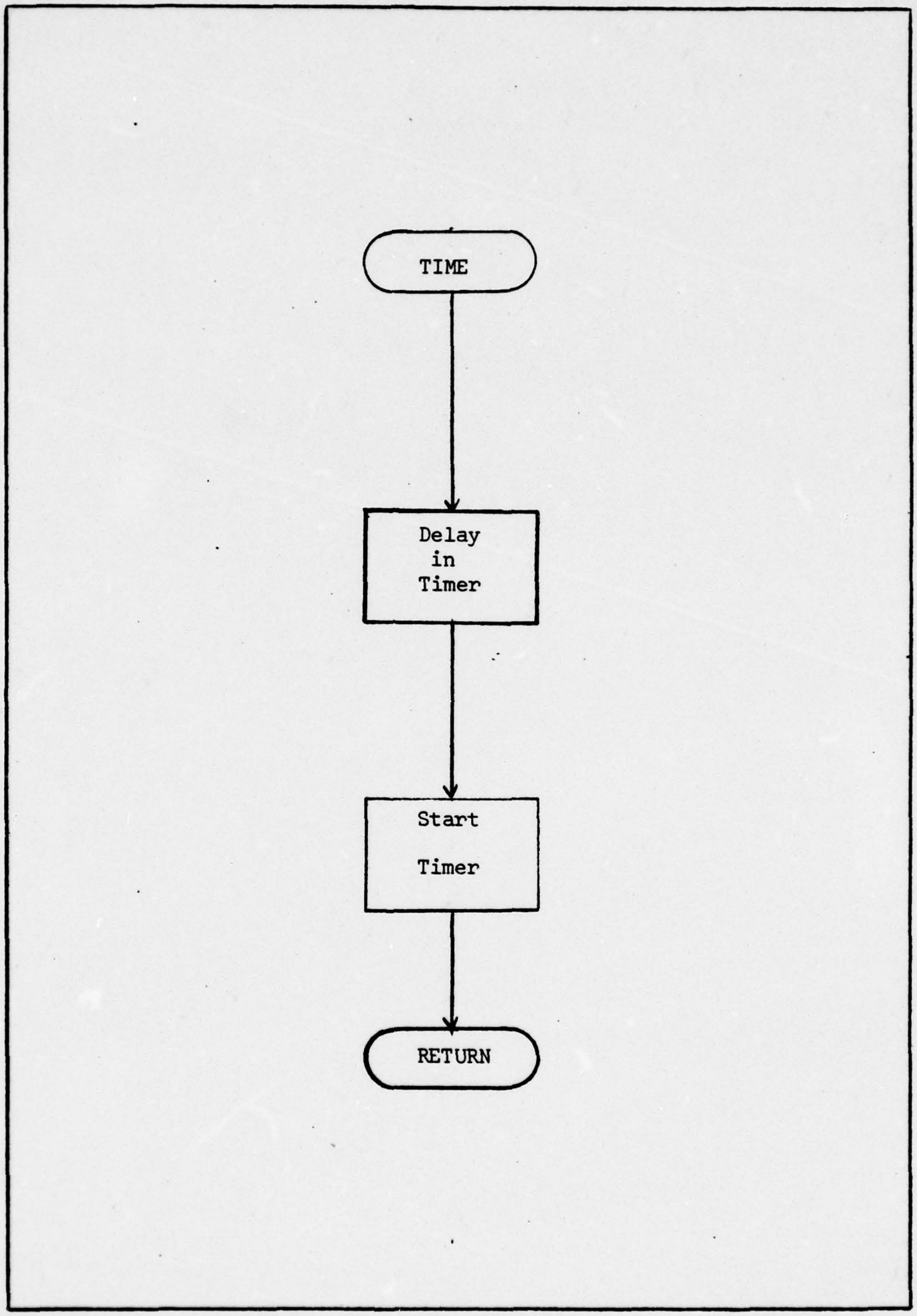


Figure IV-9. Time Flow Chart

Subtracting this from the desired 3253.7 leaves 35.4 microseconds to be accounted for. Inserting 8 NOP (no operation) instructions in the TIME module before starting the timer produces a delay of 33.5 microseconds, leaving a deviation of only 1.9 from the desired 3333.3 microseconds. This deviation over the 11 bits transmitted per character is of no consequence. The timer on the 8035 counts up so the complement of 18, E7 will be used when the TIME module is called.

The code for the Print module is located in the sections labeled PRINT through TIME. The messages referenced in the module are located after the TIME module in the exact locations addressed by the instructions referencing those messages.

Clear Accounts. The Clear Accounts module sets all the account totals to zero so a careful confirmation of the command is required. In addition to the first digit (type code) being a 5, the last 6 digits must be 505050. The first section of code simply confirms that this combination has been input. The remainder of the code simply starts at location C05 and sets every byte in the RAM to 00. On completion of the operation, registers R4 and R5 are cleared and the Display module is called to place 0000 on the display to signal completion. The code for this module is contained in the sections labeled CLEAR through HLT3.

Code. The Code module consists of taking the contents of registers R4 and R5 and storing it in locations C00 and C01 and then calling the Display module to display it. The code for this module is contained in the sections labeled CODE through HLT1.

Cost. The Cost module is the same as the Code module except the contents of register R5 are stored in location C02 and then displayed. The code for this module is in the sections labeled COST through HLT2.

Enable. The Enable module is entered through the Decode User Number module after the User Number has been determined to be valid. The address received from the Decode User Number module is normalized to the starting location of the RAM by adding 0C to the page number in register R7. The address is then checked to see that it is above the C05 starting location. The account is then fetched and if the total is found to be $99 the Enable module is aborted. The cost per copy is fetched from RAM and placed in register R6, the Xerox machine is enabled (0 in bit 2 of port 5), and the interrupt steering logic is set to recognize copy pulses (0 in bit 1 of port 5). The accumulator is set to send execution to the Count module on receiving an interrupt and control is returned to the Executive. The code for this module is contained in the sections labeled ENABLE through ERRO.

Count. The Count module simply adds the copy cost, in R6 to the account total, stores the total, and displays it. If adding the copy cost makes the account total $99 the Xerox machine is disabled by jumping to RESET. Before returning to the Executive module the accumulator is set to send execution to the Count module on receiving an interrupt. The code for this module is contained in the sections labeled COUNT through OK4.

Fetch Account. The Fetch Account module is entered with the account number to be fetched in registers R2 and R3. This register pair is left shifted to multiply by 2, since accounts take 2 locations each, and the least significant byte is placed in R0. The accounts start at location C05 so C03 is added to the R2 and R0 register pair. The first byte of the account total is fetched and put in register R6, the address is incremented and the second byte of the total is fetched and left in the accumulator for the return. The code for this module is contained in the sections labeled FETCH.

Store Account. The Store Account module is responsible for storing the account totals formed by the Increment, Decrement, and Set Account modules. The module is entered with the address of the least significant byte of the account in R2 and R0 and port 2 already latched to the proper page in the RAM. The account total to be stored is in registers R4 and R5. After the least significant byte of the total is stored, 1 is subtracted from the address by complementing the address, adding 1, and

complementing the result. The most significant byte of the account total is stored and the Display module is called to display the new total. The code for this module is contained in the sections labeled STORE and BACK.

Display. The Display module displays the contents of registers R4 and R5 as 4 hexadecimal digits. The first portion of this module clears the display and sets the display mode as desired (Ref 5:7-117). The module then masks off each digit in turn and gets the appropriate display code from the GETCD portion of the module. The display codes for the hexadecimal digits are stored in contiguous locations and the proper one is fetched by adding the digit to the address of the code for zero (DISCD) and doing an indirect fetch from memory. The display codes are formed by refering to the 7 segment display in the 8048CC User Manual and placing a 1 in the bits controlling the desired segments for each digit.

The section of code labeled COMMAND is used to send commands to the 8279 Keyboard Interface. It writes the command to the 8279 control location (FFE) and then continously reads the status word until it indicates the display is available (Ref 5:7-124). This routine is also used by the Executive though no connection is shown between the Display module and the Executive. The same is true of the OUT section of code which is used by the ECHO section of code in the Read module.

The Display module code is contained in the sections labeled COMMAND through DIS and the sections labeled DISCD through OUT.

Summary

The most important aspect of the design effort for the Xerox Controller software was the module specification phase. Once the bubble chart and structure chart of the program were complete the task reduced to the design of simple modules. Flow charts were needed in only a few places where control flow complexity required them. In most of the modules a precise description of the function was all that was needed to proceed directly to coding the module. The next chapter will present the design of the Accounts Program which is the other major software subsystem.

V ACCOUNTS PROGRAM

Introdulction

The Accounts Program is a record keeping program which is responsible for supplying the treasurer with a means for keeping track of all the authorized Xerox account users and the identification information necessary to bill them. As with the Xerox Controller program, Structured Design (SD) is used for the design of the program. The design is divided into a module specification phase and a module design phase. The source listing for the program is in Appendix B. The program is interactive so that the user need not have extensive experience to use it. FORTRAN was chosen for coding because all AFIT Engineering Students are familiar with it and could modify the program if desired. The following two sections describe the Module Specification and Module Design.

Module Specification

The bubble chart of Figure V-1 is derived from the requirements for the Accounts Program from Figure II-7. The determination of the afferent and efferent branches and the central transforms of the program was somewhat unclear and arbitrary. The Update File bubble takes a file from one of two locations and performs the updating required, so the branches which produced the files to be operated on are considered afferent branches and the Update File bubble and the transactions which follow it are considered central transforms. The braces which face left and break the New File and

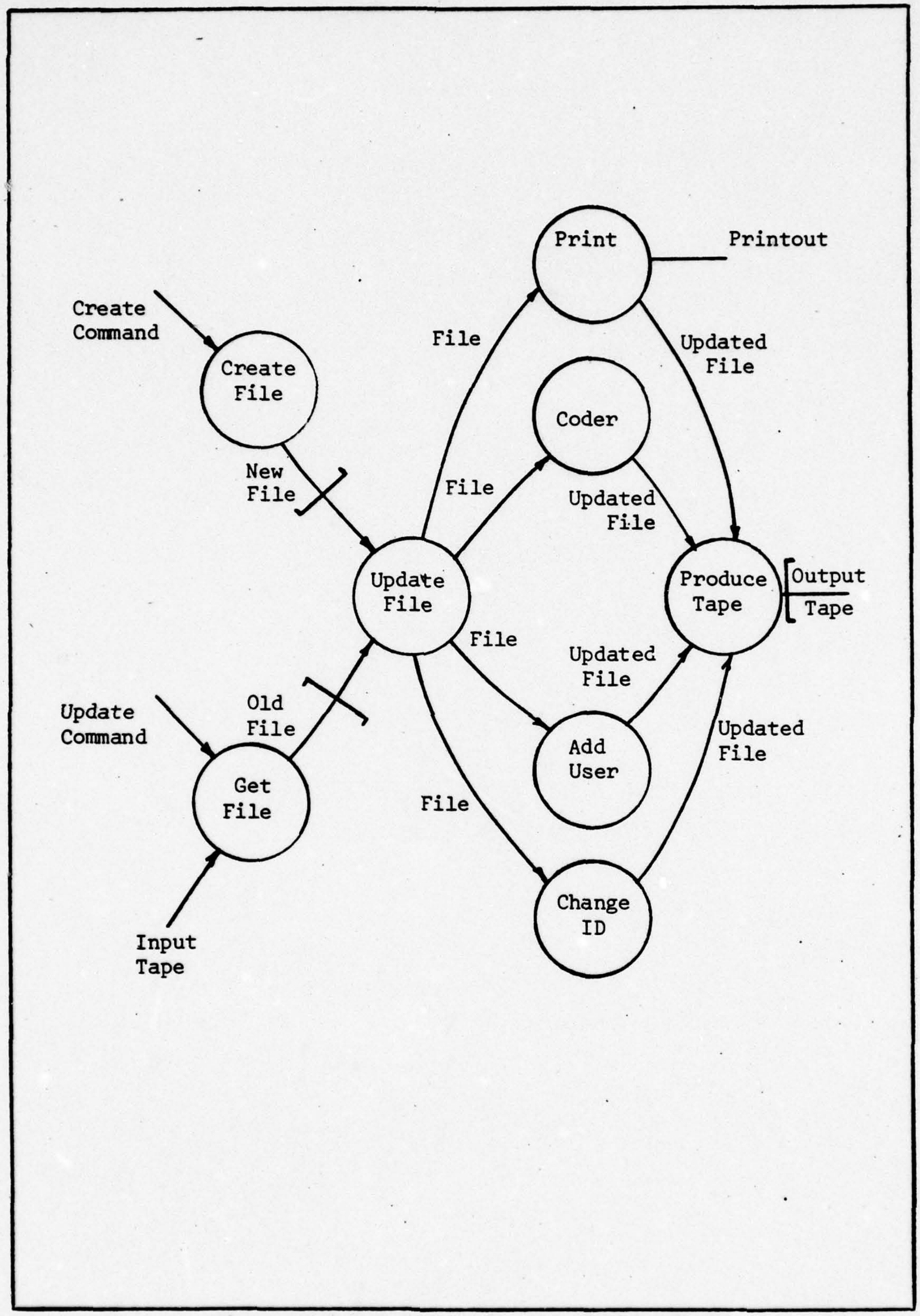


Figure V-1. Accounts Program Bubble Chart

Old File arrows show the afferent branches. The single efferent branch is indicated by the brace breaking the Output Tape arrow. The multiplicity of the arrows labeled with File is due to the nature of this program. All the bubbles on the chart operate on a common database. This common database is contained in program common storage and contains the code information and 500 account files. The account files contain User Identification (30 characters), Class designation (10 characters), and User Number (6 hexadecimal digits).

Using the bubble chart and the database description, the structure chart of Figure V-2 is produced. Some of the bubble names are not used in going from the bubble chart to the structure chart so that more functionally descriptive names could be used to name the FORTRAN modules. The bubbles labeled Get File, Add User, Change ID, and Produce Tape when transferred to the structure chart become BUFIN, NEW, INS, and ENDIT respectively. There is also a connection in the structure chart which does not show up in the bubble chart. In the process of creating a file the CODER routine is called to set up the initial encryption code. Unlike the CODER call in the Update mode, when CODER is called from the Create mode no information is passed in the call and a new code is always produced. The bubble Create File is assumed to contain all the utilities necessary to actually create the file. The CODER routine is considered a utility function which does not need to be reproduced in the Create mode and hence the cross branch connection in the structure chart.

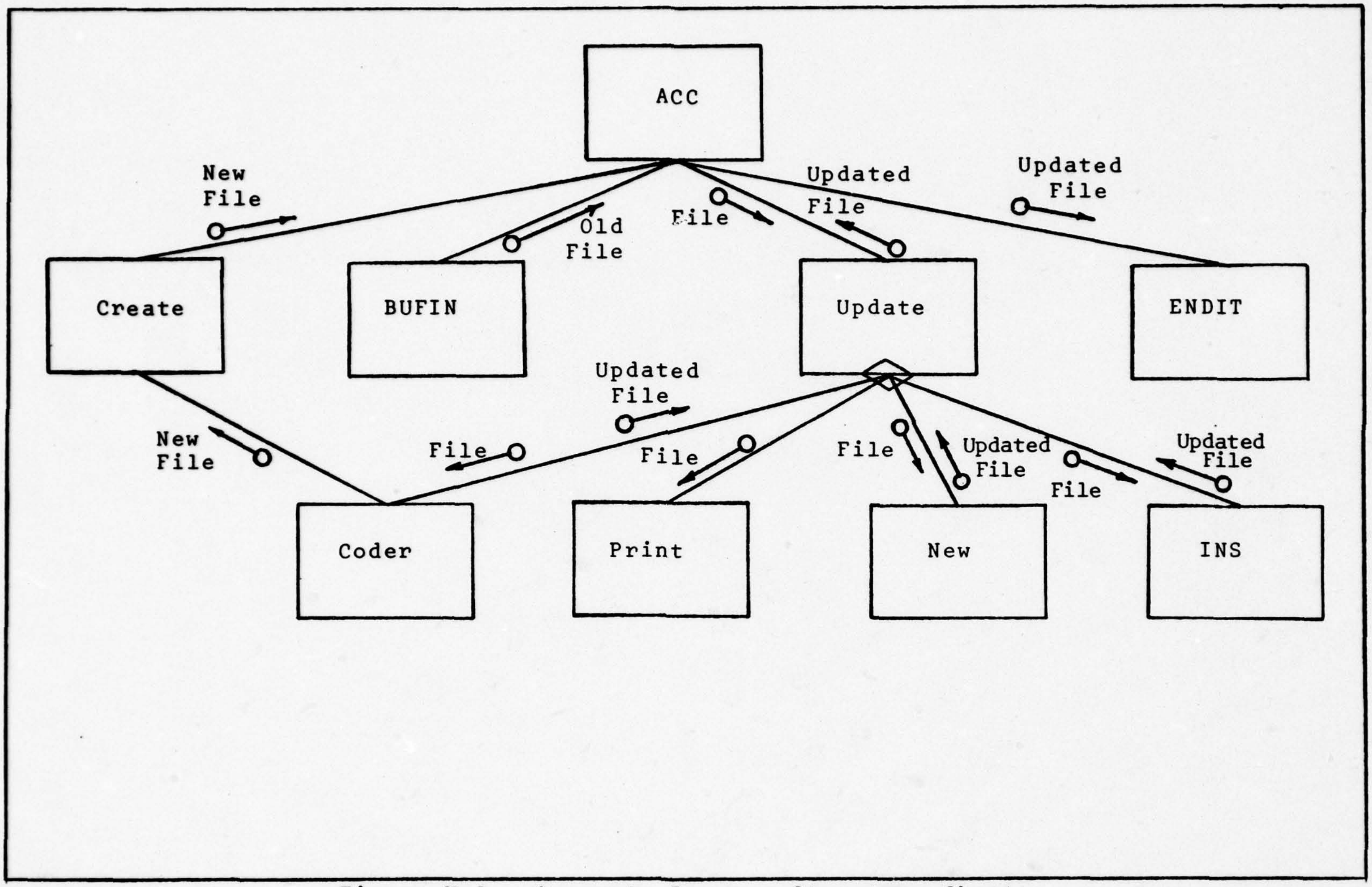


Figure V-2. Accounts Program Structure Chart

In the Module Design sections which follow, some of the functions may be further factored but the structure chart and the bubble chart presented in this section are complete enough to give a full understanding of the program.

Module Design

This program is interactive and reference to the instructions to be printed out and the source code will give the reader an easy understanding of the operation of each module. Since the program has been factored to simple modules for design, flow charts and other design aids are not necessary. For all the module descriptions which follow, IACC(I,J) is the account information file where the parameter I refers to the account number and the J parameter refers to the 5 fields in each account. The first 3 fields are the User Identification (30 characters), the next is the Class Designation (10 characters) and the last is the User Number.

ACC. The ACC module is the executive module of the program. The TAPE1, TAPE2, and TAPE3 specified in the Program statement are the Input Tape, Output Tape, and Print Tape respectively. The Print Tape is simply a local file which can be sent to the high speed printer to eliminate long time consumming printouts on the timesharing terminal. This module interactively asks the user what he wishes to do and then calls the appropriate modules.

CREATE. The CREATE module is responsible for creating an initial file. The first operation performed is to call the IZERO routine which simply initializes all account and code information to zero. The CODER routine is then called to produce the coding information on the file. The module interactively asks the user for account information for each successive account until the user indicates that he wants to discontinue input or until the 500 account spaces are used. A return is then made to the ACC module. This module as well as the IZERO module are self explanatory.

UPDATE. The UPDATE module is a transaction center which calls PRINT, CODER, NEW, INS, or ENDIT depending on the user input commands. This process is recursive so that when one of these routines is complete the module will request another command. The module is exited when the user responds with END when a command is requested and the ENDIT routine is entered to produce the output tape. This module simply selects from among the available functions and the source code is self explanatory.

BUFIN. The BUFIN module uses the FORTRAN Buffer In statement to read in a previously existing file to be used in the UPDATE routine. This preexisting file must be contained in a local file named TAPE1 for this module to read it. The alternate return, IERR, is used to indicate an error in reading the file(for example, tape not rewound or not available).

ENDIT. The ENDIT module is responsible for putting the file on TAPE2 using the FORTRAN Buffer Out statement. The Print File (TAPE3) and the Old File (TAPE1) are rewound. As in the previous module, the alternate return, IERR, is used to indicate a tape error.

PRINT. The PRINT module prints to the Print Tape the code used for encryption, the decode key to be put into the Xerox Controller, and a heading for the account information to be printed. The user is then asked if only one Class Designation accounts are desired on the printout. If a single classes accounts are to be printed those accounts with the desired Class Designation are placed on the printout, otherwise all non empty accounts are printed. The Print Tape is rewound. only when ENDIT is executed so multiple printouts can be placed consecutively on the tape.

NEW. The NEW module is responsible for adding new accounts to an existing file. The module first searches for the first empty position in the file and then allows the user to enter the account information and class designation. The module is repeated until the user indicates that he no longer wishes to enter any new accounts or until the 500 positions available sre full. When no empty position is found a message indicating that fact is output and the UPDATE routine is entered for possible deletion of unneeded accounts.

INS. The INS (inspect) module takes a 3 digit hexadecimal account number, fetches, and displays the information in that account which can then be deleted, changed, or left unchanged.

CODER and IENCRY. The CODER module and the IENCRY function control the forming of an encryption key and a corresponding decryption key, and the encryption of the account numbers to produce User Numbers. These portions of the program and the algorithm which they implement are contained in Appendix Z and because of their sensitive nature are releasable only by the author.

Summary

With the initial design and module specification done carefully and completely, the module designs reduce to a simple coding exercise. The major portion of the time spent in module design was devoted to message formatting for the interactive messages and the printout of the accounts.

VI SYSTEM TESTING

Introduction

The testing philosophy used in verifying the operation of the hardware and software of the Xerox Controller System was a bottom up approach. The system was designed top down which led to well defined functions which could be simply implemented and tested. As each module was designed, a simple but specific test procedure or test data set was used to verify the operation. The testing is covered in three sections, Hardware Testing, Xerox Controller Software Testing, and Accounts Program Testing.

Hardware Testing

The hardware testing is covered in this section in the same order as the hardware design and referring back to the design in Chapter III is helpful in understanding the testing.

The Memory Power Backup circuit shown in Figure 3-1 was tested simply by repeatedly interrupting the power to the Xerox Controller and confirming the contents of the RAM to be sure no change occurred. In the original design the diode D1 was not included and when a power interruption occurred, instead of relay R5 opening and disconnecting the remainder of the circuit board from the RAM section, the battery held the relay closed and tried to supply power to the entire board. After diode D1 was added the circuit operated as predicted and the contents of the RAM were maintained during numerous power off transfers between the library and the laboratory.

The Machine Enable Control, Figure III-2, was initially tested using an ohmmeter to check proper contact positioning under software control. When connected to the Xerox machine, switching to the enable position caused unpredictable jumping to various places in the memory to resume execution. This problem could not be duplicated on the bench until the actuating current of a 110 volt ac relay was fed through the contacts. The actuating current was measured and found to be 80 milliamps which is well below the rated current capability of the on board relays (4 amps at 220 volts). Using an oscilloscope attached to the 5 volt power supply, oscillations from 4 to 6 volts were detected upon relay actuation. No defective components (relay, transistor, or diode) could be found in the relay circuit and consultation with Imsai will be required to isolate the problem.

The Copy Pulse Buffer circuit was tested in two separate phases. The Interrupt Pulse circuit of Figure III-6 was tested first to assure the proper pulse width and immunity from multiple trigering by using a pulse generator and oscilloscope. The Pulse Shaping circuit in Figure III-5 was originally designed as a resistor divider and transistor driver combination. This circuit was connected to the Interrupt Pulse Circuit and to the copy pulse line from the Xerox machine. The Pulse Shaping circuit triggered the Interrupt Pulse circuit as desired when a copy pulse was received, however the circuit was also being triggered by the operation of the print light and cooling fan in the Xerox machine. On careful observation of the copy pulse received from the Xerox, the noise spike shown in Figure III-3 was detected and found to correspond to the false triggering of the Interrupt Pulse circuit. The capacitors were added to the circuit to filter the unwanted signals and subsequently a test was performed using hand counting to verify controller counting for a

total of 500 copies without a single miscount.

The Interrupt Steering circuit of Figure 3-7 was initially tested by simply running continuity tests on the lines from the Keyboard Interface $\overline{\mathrm{INT}}$ line and the Copy Pulse Buffer circuit to the computer $\overline{\mathrm{INT}}$ input while bit 1 of port 5 was switched between 1 and 0 under software control. After the Xerox Controller program was placed in the EPROM the operation was further confirmed during software testing.

The Keyboard circuit in Figure 3-8 was tested statically by applying a zero to the S0 and S1 lines in turn and observing the R output lines while depressing the keyboard buttons. After the circuit was connected to the computer board the operation was further confirmed by the use of the diagnostic program present in the monitor program (Imp 48) designed to test the keyboard and display.

The User/Control Switch circuit was tested using a simple software routine utilizing the JT1 instruction which senses the input value on the T1 line to which the circuit is connected.

<u>Xerox Controller Software Testing</u>

The Xerox Controller Software testing was accomplished using the versatile Imp 48 monitor program (Ref   ) supplied with the 8048CC computer. The RAM was expanded to 2k bytes, by adding eight 2102 integrated circuits, to allow testing program modules in the RAM without using the memory space intended for the accounts. A path testing technique was employed to test each module. The basic function of each module was tested using input data which forced every path through the module to be exercised. The break function of the monitor

program allowed the execution of a few instructions at a time with the capability to examine all registers and memory locations between executions of adjacent sections of code. The simplicity of the modules which were created by the top down design made the actual testing extremely easy. Most of the errors detected in testing were coding errors, usually involving incorrect register addressing. During the testing of the Increment Account module, an error in the execution of the DAA (decimal adjust accumulator) instruction was detected. The 8035 could not recognize and adjust a hexadecimal A or B. After a replacement for the 8035 was obtained, the problem was remedied. Another problem which was encountered during the testing of the Read module (which must recognize multiple interrupts from the keyboard) was the inability of the program to respond to an interrupt after returning from the interrupt service routine (the Read module in this case). This problem was due to not using the RETR (return and restore) on return from an interrupt service routine as required. If the RET (return) instruction is used, as was incorrectly done in the Read module, further interrupts will not be recognized even if the EN I (enable interrupts) instruction is executed. Replacing the incorrect instructions solved the problem and no further problems were encountered in the module testing. The entire program was placed in the RAM and all the module interfaces were tested and found to operate correctly. The program was inserted into the EPROM using the procedure explained in Appendix C.

Accounts Program Testing

The Accounts Program testing reduced to eliminating compilation errors (syntax errors) and refining the formats of the interactive messages and the print tape. Testing the overall operation of the program was accomplished by producing a sample file and trying all the possible operations on it. The path testing technique was used to be sure that every path performed as expected. The program was reexecuted using the output tape from the first execution to confirm that the program could take a previously produced tape and perform any of the desired operations on it.

Summary

The testing of the Xerox Controller System, because of the careful top down design approach utilized throughout, reduced to simple and straightforward module testing. In the case of the software, the modules were tested using the path testing technique to assure that every module did exactly the intended function and nothing else. With the exception of the problem noted in the discussion of the Machine Enable circuit, the testing was completed quickly and efficiently.

VI RESULTS AND CONCLUSIONS

Introduction

The results of this design study will be presented as compliance or noncompliance with the objectives listed in Chapter I. The appropriate conclusions which can be drawn from these results will be discussed and recommendations for any continued effort will be presented.

Results

The results in terms of the objectives listed in Chapter I are as follows:

1. The 8048CC SBC and the necessary peripheral hardware was selected and designed to handle maintenance of user accounts and control the operation of the Xerox machine. Though part of the Xerox Controller circuitry is not operating, this is not considered a design deficiency, but rather a probable component failure.
2. A secure procedure has been implemented for uniquely identifying a valid Xerox account user. The security of this system is presented in Appendix Z which is sensitive because of its nature and can only be obtained from the author. It can be stated here without comprimising the security of the system that the probability of entering a random number and gaining access to an account is only .00006.

3. The Xerox Controller software was designed and implemented to perform the functions stated in the objectives. The operators manual for the Xerox Controller is contained in Appendix D.
4. The Accounts Program was designed and implemented to maintain a record of all valid users and the identification information necessary to allow the treasurer to identify and bill users for usage. The user manual for the Accounts Program is contained in Appendix E.
5. The test and validation has only been partially met since there is still a problem with the Machine Enable circuit.

Conclusions

The Xerox Controller has been designed using a top down approach. The implementation is complete except for what is considered a solvable problem with the Machine Enable circuit. This problem is detailed in the System Testing chapter.

Recommendations

A number of follow on studies are possible to enhance the accounting which must be accomplished. If a method of transmitting data from the Xerox Controller directly to the CYBER system can be devised, the entire accounting procedure could be automated by including it in the Accounts Program. Another approach would be to produce a paper or magnetic tape output which could then be transmitted to the CYBER system.

Bibliography

1. Conway, John. "Computers and Peripherals-The Day of the Micropriced System," EDN 13: 20 July 1977.

2. Diffie, W. and M. Hellman, "New Directions in Cryptography," IEEE Transactions on Information Theory __:November 1976.

3. Imsai Manufacturing Corp. 8048 Control Computer User Manual. San Leandro: 1977.

4. Intel Corp. Data Catalog 1977. Santa Clara: 1977.

5. Intel Corp. MCS-48 Microcomputer User's Manual. Santa Clara: 1977.

6. Miller, Capt Peter Instructor, Air Force Institute of Technology Electrical Engineering Department (personal interviews) 1977.

7. Osborne, Adam, et al. An Introduction to Microcomputers Volume II. Berkeley: Adam Osborne and Associates,1977.

8. Rivest, Ronald L. et al. On Digital Signatures and Public-Key Cryptosystems. Report. Cambridge: 1977.

9. Softech, Inc. An Introduction to SADT, Structured Analysis Design Technique. Waltham: November 1976.

10. Softech, Inc. Structured Analysis Reader Guide. Waltham: May 1975.

11. Texas Instruments Inc. The TTL Data Book. Dallas: 1973.

12. Yourdan, Ed and Larry L. Constantine, Structured Design. New York: Yourdan Inc, 1975.

A XEROX CONTROLLER SOURCE LISTING

LABEL	LOC	OP	ASSEMBLY
RESET	000	00	NOP
	001	0418	JMP RST
INT	003	00	NOP
	004	4691	JNT1 USR
	006	C639	JZ FIRST
	008	1241	JB0 SECOND
	00A	3247	JB1 THIRD
	00C	524E	JB2 4TH
	00E	7255	JB3 5TH
	010	925C	JB4 6TH
	012	B263	JB5 7TH
	014	D26A	JB6 8TH
	016	F271	JB7 TEST
RST	018	15	DIS I
	019	230F	MOV A,#0F
	01B	3D	MOVD P5,A
	01C	230F	MOV A,#0F
	01E	3A	OUTL P2,A
	01F	B9FF	MOV R1,#FF
	021	2324	MOV A,#24
	023	3499	CALL COMMAND
	025	23D1	MOV A,#D1
	027	3499	CALL COMMAND
	029	2390	MOV A,#90
	02B	3499	CALL COMMAND
	02D	2340	MOV A,#40
	02F	3499	CALL COMMAND
	031	B8FE	MOV R0,#FE
	033	27	CLR A
	034	62	MOV T,A
	035	55	STRT T
STBY	036	05	EN I
	037	0436	JMP STBY
FIRST	039	148B	CALL ECHO
	03B	5307	ANL A,#07
	03D	AF	MOV R7,A
	03E	2301	MOV A,#SECOND
	040	93	RETR
SECOND	041	148B	CALL ECHO
	043	AA	MOV R2,A
	044	2302	MOV A,#THIRD
	046	93	RETR
THIRD	047	148B	CALL ECHO
	049	47	SWAP A
	04A	AB	MOV R3,A
	04B	2304	MOV A,#4TH
	04D	93	RETR
4TH	04E	148B	CALL ECHO
	050	4B	ORL A,R3
	051	AB	MOV R3,A
	052	2308	MOV A,#5TH
	054	93	RETR

LABEL	LOC	OP	ASSEMBLY
5TH	055	148B	CALL ECHO
	057	47	SWAP A
	058	AC	MOV R4,A
	059	2310	MOV A,#6TH
	05B	93	RETR
6TH	05C	148B	CALL ECHO
	05E	4C	ORL A,R4
	05F	AC	MOV R4,A
	060	2320	MOV A,#7TH
	062	93	RETR
7TH	063	148B	CALL ECHO
	065	47	SWAP A
	066	AD	MOV R5,A
	067	2340	MOV A,#8TH
	069	93	RETR
8TH	06A	148B	CALL ECHO
	06C	4D	ORL A,R5
	06D	AD	MOV R5,A
	06E	2380	MOV A,#TEST
	070	93	RETR
TEST	071	EF75	DJNZ R7,T2
	073	240A	JMP INC
T2	075	EF79	DJNZ R7,T3
	077	242B	JMP DEC
T3	079	EF7D	DJNZ R7,T4
	07B	2448	JMP SET
T4	07D	EF81	DJNZ R7,T5
	07F	44F5	JMP PRINT
T5	081	EF85	DJNZ R7,T6
	083	244C	JMP CLEAR
T6	085	EF89	DJNZ R7,T7
	087	247E	JMP CODE
T7	089	248C	JMP COST
ECHO	08B	80	MOVX A,@RO
	08C	530F	ANL A,#OF
	08E	14E7	CALL OUT
	090	83	RET
USR	091	C69F	JZ USR1
	093	12A8	JBO USR2
	095	32AF	JB1 USR3
	097	52B6	JB2 USR4
	099	72BD	JB3 USR5
	09B	92C4	JB4 USR6
	09D	44D0	JMP COUNT
USR1	09F	BFFF	MOV R7,#FF
	0A1	14CA	CALL READ
	0A3	47	SWAP A
	0A4	AA	MOV R2,A
	0A5	2301	MOV A,#USR2
	0A7	93	RETR

LABEL	LOC	OP	ASSEMBLY
USR2	0A8	14CA	CALL READ
	0AA	4A	ORL A,R2
	0AB	AA	MOV R2,A
	0AC	2302	MOV A,#USR3
	0AE	93	RETR
USR3	0AF	14CA	CALL READ
	0B1	47	SWAP A
	0B2	AB	MOV R3,A
	0B3	2304	MOV A,#USR4
	0B5	93	RETR
USR4	0B6	14CA	CALL READ
	0B8	4B	ORL A,R3
	0B9	AB	MOV R3,A
	0BA	2308	MOV A,#USR5
	0BC	93	RETR
USR5	0BD	14CA	CALL READ
	0BF	47	SWAP A
	0C0	AC	MOV R4,A
	0C1	2310	MOV A,#USR6
	0C3	93	RETR
USR6	0C4	14CA	CALL READ
	0C6	4C	ORL A,R4
	0C7	AC	MOV R4,A
	0C8	24C7	JMP DECODE
READ	0CA	80	MOVX A,@R0
	0CB	530F	ANL A,#0F
	0CD	2F	XCH A,R7
	0CE	90	MOVX@R0,A
	0CF	2F	XCH A,R7
	0D0	83	RET
DISCD	0D1	3F	0
	0D2	06	1
	0D3	5B	2
	0D4	4F	3
	0D5	66	4
	0D6	6D	5
	0D7	7D	6
	0D8	07	7
	0D9	7F	8
	0DA	67	9
	0DB	77	A
	0DC	7C	B
	0DD	39	C
	0DE	5E	D
	0DF	79	E
	0E0	71	F
GETCD	0E1	D5	SEL RB1
	0E2	03D1	ADD A,#DISCD
	0E4	A3	MOVP A,@A
	0E5	C5	SEL RB0
	0E6	83	RET

LABEL	LOC	OP	ASSEMBLY
OUT	0E7	AE	MOV R6,A
	0E8	14E1	CALL GETCD
	0EA	90	MOVX@R0,A
	0EB	FE	MOV A,R6
	0EC	83	RET
FETCH	0ED	97	CLR C
	0EE	FB	MOV A,R3
	0EF	F7	RLC A
	0F0	A8	MOV R0,A
	0F1	FA	MOV A,R2
	0F2	F7	RLC A
	0F3	AA	MOV R2,A
	0F4	F8	MOV A,R0
	0F5	0303	ADD A,#03
	0F7	A8	MOV R0,A
	0F8	FA	MOV A,R2
	0F9	130C	ADDC A,#0C
	0FB	AA	MOV R2,A
	0FC	3A	OUTL P2,A
	0FD	80	MOVX A,@R0
	0FE	AE	MOV R6,A
	0FF	F8	MOV A,R0
	100	0301	ADD A,#01
	102	A8	MOV R0,A
	103	FA	MOV A,R2
	104	1300	ADDC A,#00
	106	3A	OUTL P2,A
	107	AA	MOV R2,A
	108	80	MOVX A,@R0
	109	83	RET
INC	10A	14ED	CALL FETCH
	10C	6D	ADD A,R5
	10D	57	DAA
	10E	AD	MOV R5,A
	10F	FE	MOV A,R6
	110	7C	ADDC A,R4
	111	57	DAA
	112	AC	MOV R4,A
	113	E617	JNC STORE
	115	0400	JMP RESET

LABEL	LOC	OP	ASSEMBLY
STORE	117	FD	MOV A,R5
	118	90	MOVX@RO,A
	119	F8	MOV A,RO
	11A	37	CPL A
	11B	0301	ADD A,#01
	11D	37	CPL A
	11E	A8	MOV RO,A
	11F	FA	MOV A,R2
	120	37	CPL A
	121	1300	ADDC A,#00
	123	37	CPL A
	124	3A	OUTL P2,A
	125	FC	MOV A,R4
	126	90	MOVX@RO,A
	127	349E	CALL DIS
BACK	129	2429	JMP BACK
DEC	12B	14ED	CALLFETCH
	12D	BB66	MOV R3,#66
	12F	6B	ADD A,R3
	130	37	CPL A
	131	AF	MOV R7,A
	132	FE	MOV A,R6
	133	6B	ADD A,R3
	134	37	CPL A
	135	AE	MOV R6,A
	136	FF	MOV A,R7
	137	6D	ADD A,R5
	138	57	DAA
	139	AD	MOV R5,A
	13A	FE	MOV A,R6
	13B	7C	ADDC A,R4
	13C	57	DAA
	13D	F65A	JC ERR
	13F	6B	ADD A,R3
	140	37	CPL A
	141	AC	MOV R4,A
	142	FD	MOV A,R5
	143	6B	ADD A,R3
	144	37	CPL A
	145	AD	MOV R5,A
	146	2417	JMP STORE
SET	148	14ED	CALL FETCH
	14A	2417	JMP STORE

LABEL	LOC	OP	ASSEMBLY
CLEAR	14C	BE50	MOV R6,#50
	14E	FB	MOV A,R3
	14F	DE	XRL A,R6
	150	965A	JNZ ERR
	152	FC	MOV A,R4
	153	DE	XRL A,R6
	154	965A	JNZ ERR
	156	FD	MOV A,R5
	157	DE	XRL A,R6
	158	C65C	JZ OK
ERR	15A	0400	JMP RESET
OK	15C	B805	MOV R0,#05
	15E	27	CLR A
	15F	BE0C	MOV R6,#0C
	161	2E	XCH A,R6
FLIP	162	3A	OUTL P2,A
	163	2E	XCH A,R6
LOOP	164	90	MOVX@R0,A
	165	18	INC R0
	166	28	XCH A,R0
	167	C66C	JZ PAGE
	169	28	XCH A,R0
	16A	2464	JMP LOOP
PAGE	16C	1E	INC R6
	16D	2E	XCH A,R6
	16E	D310	XRL A,#10
	170	C676	JZ END
	172	D310	XRL A,#10
	174	2462	JMP FLIP
END	176	BC00	MOV R4,#00
	178	BD00	MOV R5,#00
	17A	349E	CALL DIS
HLT3	17C	247C	JMP HLT3
CODE	17E	B800	MOV R0,#NBITS
	180	230C	MOV A,#0C
	182	3A	OUTL P2,A
	183	FC	MOV A,R4
	184	90	MOVX@R0,A
	185	18	INC R0
	186	FD	MOV A,R5
	187	90	MOVX@R0,A
	188	349E	CALL DIS
HLT1	18A	248A	JMP HLT1
COST	18C	B802	MOV R0,#COSTAD
	18E	230C	MOV A,#0C
	190	3A	OUTL P2,A
	191	BC00	MOV R4,#00
	193	FD	MOV A,R5
	194	90	MOVX@R0,A
	195	349E	CALL DIS
HLT2	197	2497	JMP HLT2

LABEL	LOC	OP	ASSEMBLY
COMMAND	199	91	MOVX@R1,A
L1	19A	81	MOVX A,@R1
	19B	F29A	JB7 L1
	19D	83	RET
DIS	19E	230F	MOV A,#0F
	1A0	3A	OUTL P2,A
	1A1	B9FF	MOV R1,#FF
	1A3	23D1	MOV A,#D1
	1A5	3499	CALL COMMAND
	1A7	2390	MOV A,#90
	1A9	3499	CALL COMMAND
	1AB	C9	DEC R1
	1AC	FC	MOV A,R4
	1AD	47	SWAP A
	1AE	530F	ANL A,#0F
	1B0	14E1	CALL GETCD
	1B2	91	MOVX@R1,A
	1B3	FC	MOV A,R4
	1B4	530F	ANL A,#0F
	1B6	14E1	CALL GETCD
	1B8	91	MOVX@R1,A
	1B9	FD	MOV A,R5
	1BA	47	SWAP A
	1BB	530F	ANL A,#0F
	1BD	14E1	CALL GETCD
	1BF	91	MOVX@R1,A
	1C0	FD	MOV A,R5
	1C1	530F	ANL A,#0F
	1C3	14E1	CALL GETCD
	1C5	91	MOVX@R1,A
	1C6	83	RET
ENABLE	29B	FF	MOV A,R7
	29C	030C	ADD A,#0C
	29E	3A	OUTL P2,A
	29F	AA	MOV R2,A
	2A0	D30C	XRL A,#0C
	2A2	96AB	JNZ EN1
	2A4	F8	MOV A,R0
	2A5	97	CLR C
	2A6	67	RRC
	2A7	97	CLR C
	2A8	67	RRC
	2A9	C6CE	JZ ERR0
EN1	2AB	80	MOVX A,@R0
	2AC	AC	MOV R4,A
	2AD	D399	XRL A,#MAX$
	2AF	C6CE	JZ ERR0
	2B1	18	INC R0
	2B2	F8	MOV A,R0
	2B3	96B8	JNZ OK1
	2B5	1A	INC R2
	2B6	FA	MOV A,R2
	2B7	3A	OUTL P2,A

LABEL	LOC	OP	ASSEMBLY
OK1	2B8	80	MOVX A,@RO
	2B9	AD	MOV R5,A
	2BA	349E	CALL DIS
	2BC	F8	MOV A,RO
	2BD	96CO	JNZ OK2
	2BF	CA	DEC R2
OK2	2C0	C8	DEC RO
	2C1	230C	MOV A,#0C
	2C3	3A	OUTL P2,A
	2C4	B902	MOV R1,#COSTAD
	2C6	81	MOVX A,@R1
	2C7	AE	MOV R6,A
	2C8	2309	MOV A,#09
	2CA	3D	MOVD P5,A
SELF	2CB	2320	MOV A,#COUNT(20H)
	2CD	93	RETR
ERRO	2CE	0400	JMP RESET
COUNT	2D0	FD	MOV A,R5
	2D1	6E	ADD A,R6
	2D2	57	DAA
	2D3	AD	MOV R5,A
	2D4	FC	MOV A,R4
	2D5	1300	ADDC A,#00
	2D7	57	DAA
	2D8	AC	MOV R4,A
	2D9	FA	MOV A,R2
	2DA	3A	OUTL P2,A
	2DB	FC	MOV A,R4
	2DC	90	MOVX@RO,A
	2DD	18	INC RO
	2DE	F8	MOV A,RO
	2DF	96E4	JNZ OK3
	2E1	1A	INC R2
	2E2	FA	MOV A,R2
	2E3	3A	OUTL P2,A
OK3	2E4	FD	MOV A,R5
	2E5	90	MOVX@RO,A
	2E6	F8	MOV A,RO
	2E7	96EA	JNZ OK4
	2E9	CA	DEC R2
OK4	2EA	C8	DEC RO
	2EB	FC	MOV A,R4
	2EC	D399	XRL A,#MAX$
	2EE	C6CE	JZ ERRO
	2F0	349E	CALL DIS
	2F2	2320	MOV A,#COUNT(20)
	2F4	93	RETR

LABEL	LOC	OP	ASSEMBLY
PRINT	2F5	741B	CALL HEADING
	2F7	B900	MOV R1,#00
	2F9	BB01	MOV R3,#01
FET	2FB	F9	MOV A,R1
	2FC	AA	MOV R2,A
	2FD	14ED	CALL FETCH
	2FF	AF	MOV R7,A
	300	4E	ORL A,R6
	301	C605	JZ SKIP
NONZERO	303	7434	CALL PRINAC
SKIP	305	1B	INC R3
	306	FB	MOV A,R3
	307	C611	JZ FLIPPAGE
	309	D3F4	XRL A,#F4
	30B	C614	JZ CHECKEND
	30D	D3F4	XRL A,#F4
	30F	44FB	JMP FET
FLIPPAGE	311	19	INC R1
	312	44FB	JMP FET
CHECKEND	314	F9	MOV A,R1
	315	C612	JZ FET
	317	7420	CALL TRAILER
	319	0400	JMP RESET
HEADING	31B	2392	MOV A,#HEADING ADDRESS
	31D	7425	CALL PRINTER
	31F	83	RET
TRAILER	320	23CB	MOV A,#TRAILER ADDRESS
	322	7425	CALL PRINTER
	324	83	RET
PRINTER	325	A8	MOV R0,A
PLOOP	326	A3	MOVP A,@A
	327	D307	XRL A,#07 (BEL)
	329	C633	JZ END
	32B	D307	XRL A,#07 (BEL)
	32D	7465	CALL SEROUT
	32F	18	INC R0
	330	F8	MOV A,R0
	331	6426	JMP PLOOP
END	333	83	RET

```
LABEL     LOC  OP    ASSEMBLY
PRINAC    334  F9    MOV A,R1
          335  7459  CALL ASCII
          337  FB    MOV A,R3
          338  47    SWAP A
          339  7459  CALL ASCII
          33B  FB    MOV A,R3
          33C  7459  CALL ASCII
          33E  23BE  MOV A,#SPACER ADDRESS
          340  7425  CALL PRINTER
          342  FE    MOV A,R6
          343  47    SWAP A
          344  7459  CALL ASCII
          346  FE    MOV A,R6
          347  7459  CALL ASCII
          349  232E  MOV A,#2E (.)
          34B  7465  CALL SEROUT
          34D  FF    MOV A,R7
          34E  47    SWAP A
          34F  7459  CALL ASCII
          351  FF    MOV A,R7
          352  7459  CALL ASCII
          354  23C6  MOV A,#ENDLINE
          356  7425  CALL PRINTER
          358  83    RET
ASCII     359  530F  ANL A,#0F
          35B  0330  ADD A,#30
          35D  57    DAA
          35E  00    NOP
          35F  9262  JB4 READY
          361  17    INC A
READY     362  7465  CALL SEROUT
          364  83    RET
SEROUT    365  D5    SEL RB1
          366  AB    MOV R3,A
          367  B90B  MOV R1,#0B
          369  97    CLR C
SER1      36A  166E  JTF SER2
          36C  646A  JMP SER1
SER2      36E  E675  JNC ONE
          370  230B  MOV A,#0B
          372  9F    ANLD P7,A
          373  647A  JMP SER3
ONE       375  2304  MOV A,#04
          377  8F    ORLD P7,A
          378  647A  JMP SER3
SER3      37A  97    CLR C
          37B  A7    CPL C
          37C  FB    MOV A,R3
          37D  67    RRC
          37E  AB    MOV R3,A
          37F  23E7  MOV A,#DELAY (E7)
          381  7487  CALL TIME
          383  E96A  DJNZ R1,SER1
          385  C5    SEL RB0
          386  83    RET
```

LABEL	LOC	OP	ASSEMBLY
TIME	387	62	MOV T,A
	388	00	NOP
	389	00	NOP
	38A	00	NOP
	38B	00	NOP
	38C	00	NOP
	38D	00	NOP
	38E	00	NOP
	38F	00	NOP
	390	55	STRT T
	391	83	RET

B ACCOUNTS PROGRAM SOURCE LISTING

```
      PROGRAM ACC(INPUT,OUTPUT,TAPE1=0,TAPE2=0,TAPE3)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
1     CONTINUE
      PRINT 100
100   FORMAT(* SYSTEM OPTIONS:TO ENTER CREATE ENTER CR.*/
     &* TO ENTER UPDATE ENTER UP.*/* TO EXIT ENTER EX.*//,* $*)
      READ 200,IN
200   FORMAT(R2)
      IF(IN.EQ.2RCR)CALL CREATE,RETURNS(2)
      IF(IN.EQ.2RUP)CALL BUFIN,RETURNS(4,5)
      IF(IN.EQ.2REX)GO TO 6
      PRINT 300
300   FORMAT(* IMPROPER INPUT CHECK OPTIONS AND REENTER.*)
      GO TO 1
2     CONTINUE
3     PRINT 400
400   FORMAT(* CREATE COMPLETE DO YOU WISH TO ENTER UPDATE?*
     &,//,* ENTER YES OR NO.*/,* $*)
      READ 200,IN
      IF(IN.EQ.2RYE)CALL UPDATE,RETURNS(4,5)
      IF(IN.EQ.2RNO)CALL ENDIT,RETURNS(4,5)
      PRINT 300
      GO TO 3
5     CONTINUE
      PRINT 600
600   FORMAT(* TAPE READ OR WRITE FAILURE.  IF IN CREATE*/
     &* START OVER.  IF IN UPDATE REEXECUTE WITH TAPE 1 OR 2.*)
      GO TO 6
4     CONTINUE
      PRINT 500
500   FORMAT(* RUN COMPLETE NEW FILE ON TAPE2 PRINTOUT ON TAPE3.*)
6     CONTINUE
      STOP
      END
```

```
      SUBROUTINE CREATE,RETURNS(NORM)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      DIMENSION ITEMP(4)
      DATA IEND/10HEND       /
      PRINT 100
100   FORMAT(* ENTERING THE CREATE MODE.*)
      CALL IZERO
      CALL CODER,RETURNS(4)
4     I=1
1     PRINT 200
200   FORMAT(* ENTER ACCOUNT INFORMATION UP TO 30 CHARACTERS*
     &/* OR TO EXIT CREATE ENTER END.*//,* #*)
      READ 300,(ITEMP(J),J=1,3)
300   FORMAT(3A10)
      IF(ITEMP(1).EQ.IEND)GO TO 2
      PRINT 310
310   FORMAT(* ENTER CLASS DESIGNATOR.*/* #*)
      READ 320,ITEMP(4)
320   FORMAT(A10)
      DO 3 J=1,4
      IACC(I,J)=ITEMP(J)
3     CONTINUE
      IACC(I,5)=IENCRY(2*I+3)
      I=I+1
      IF(I.LE.500)GO TO 1
      PRINT 400
400   FORMAT(* THE ACCOUNT TAPE IS FULL.  CREATE IS BEING TERMINATED.*)
      RETURN NORM
2     CONTINUE
      RETURN NORM
      END
```

```
      SUBROUTINE UPDATE,RETURNS(NORM,IERR)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
1     PRINT 100
100   FORMAT(* UPDATE ROUTINE ENTER PRINT,CODE,NEW,INSPECT OR END.*//
     &,* #*)
      READ 200,ITEMP
200   FORMAT(R1)
      IF(ITEMP.EQ.1RP)CALL PRINT,RETURNS(1)
      IF(ITEMP.EQ.1RC)CALL CODER,RETURNS(1)
      IF(ITEMP.EQ.1RN)CALL NEW,RETURNS(1)
      IF(ITEMP.EQ.1RI)CALL INS,RETURNS(1)
      IF(ITEMP.EQ.1RE)CALL ENDIT,RETURNS(2,3)
      PRINT 300
300   FORMAT(* IMPROPER ENTRY, PLEASE REENTER.*)
      GO TO 1
2     CONTINUE
      RETURN NORM
3     CONTINUE
      RETURN IERR
      END


      SUBROUTINE BUFIN,RETURNS(NORM,IERR)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      REWIND 1
      BUFFER IN(1,0) (ICOD,IACC(500,5))
      IF(UNIT(1))1,2,2
1     CALL UPDATE,RETURNS(3,2)
2     CONTINUE
      RETURN IERR
3     CONTINUE
      RETURN NORM
      END
```

```
      SUBROUTINE ENDIT,RETURNS(NORM,IERR)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      REWIND 1
      REWIND 2
      REWIND 3
      BUFFER OUT (2,0) (ICOD,IACC(500,5))
      IF(UNIT(2))1,2,2
1     CONTINUE
      REWIND 2
      RETURN NORM
2     CONTINUE
      RETURN IERR
      END
```

```
      SUBROUTINE IZERO
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      ICOD=0
      NBITS=0
      IDECOD=0
      DO 1 I=1,500
      DO 1 J=1,5
      IACC(I,J)=0
1     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE PRINT,RETURNS (NORM)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      WRITE (3,100) ICOD,IDECOD
100   FORMAT(1H ,*THE CODE USED FOR ENCRYPTION IS  *,Z7//
     &,* THE CODE INPUT TO THE CONTROLLER SHOULD BE  *
     &,1H ,2X,*18*,Z2,//
     &* ACCT (HEX)*,4X,*IDENTIFICATION*,13X,*CLASS*,5X,*CODE (HEX)*)
      PRINT 300
300   FORMAT(* DO YOU WISH TO PRINT A PARTICULAR CLASS ACCOUNTS?*
     &/* ENTER YES OR NO.*/* #*)
      READ 400,ITEMP
400   FORMAT(R2)
      IF(ITEMP.EQ.2RYE)GO TO 2
      DO 1 I=1,500
      IF(IACC(I,1).EQ.0)GO TO 1
      WRITE (3,200) I,(IACC(I,J),J=1,5)
200   FORMAT(1H ,Z3,8X,4A10,2X,Z6,//)
1     CONTINUE
      RETURN NORM
2     CONTINUE
      PRINT 500
500   FORMAT(* ENTER THE CLASS DESIGNATOR OF THE CLASS ACCOUNTS*
     &/* YOU WANT PRINTED.*/* #*)
      READ 600,ITEMP
600   FORMAT(A10)
      DO 3 I=1,500
      IF(IACC(I,4).NE.ITEMP)GO TO 3
      WRITE (3,200) I,(IACC(I,J),J=1,5)
3     CONTINUE
      RETURN NORM
      END
```

```
      SUBROUTINE NEW,RETURNS(NORM)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
3     CONTINUE
      DO 1 I=1,500
      IF(IACC(I,1).EQ.0)GO TO 2
1     CONTINUE
      PRINT 100
100   FORMAT(* NO SPACE IS AVAILABLE RETURNING TO UPDATE FOR POSSIBLE*
     &/* DELETION.*)
      RETURN NORM
2     CONTINUE
      PRINT 200,I
200   FORMAT(* ACCOUNT#  *,Z3,* IS OPEN IF YOU WISH TO ENTER*
     &/* AN ACCOUNT ENTER YES.*//,* #*)
      READ 300,ITEMP
300   FORMAT(R1)
      IF(ITEMP.NE.1RY)RETURN NORM
      PRINT 400
400   FORMAT(* ENTER UP TO 30 CHARACTER IDENTIFICATION OF ACCOUNT.*//,
     &* #*)
      READ 500,(IACC(I,J),J=1,3)
500   FORMAT(3A10)
      IACC(I,5)=IENCRY(2*I+3)
      PRINT 600
600   FORMAT(* ENTER CLASS DESIGNATOR.*/* #*)
      READ 700,IACC(I,4)
700   FORMAT(A10)
      GO TO 3
      END
```

```
      SUBROUTINE INS,RETURNS(NORM)
      COMMON ICOD,NBITS,IDECOD,IACC(500,5)
      PRINT 100
100   FORMAT(* ENTERING INSPECTION ROUTINE, ENTER ACCOUNT NUMBER.*
     &/* ENTRY MUST BE A 3 DIGIT HEX NUMBER (EX 1A7).*/* #*)
      READ 200,I
200   FORMAT(Z3)
4     PRINT 300,I,(IACC(I,J),J=1,5)
300   FORMAT(1H ,*ACCT*,10X,*IDENTIFICATION*,7X,*CLASS*,5X,
     &*CODE*/
     &,1H ,Z3,2X,4A10,2X,Z6//* IF YOU WISH TO DELETE THIS ACCOUNT*
     &* ENTER DE.*/,* IF YOU WISH TO CHANGE THE ID ENTER ID.*/
     &,* IF NO CHANGE IS DESIRED ENTER NO.*//,* #*)
      READ 400,ITEMP
400   FORMAT(R2)
      IF(ITEMP.EQ.2RDE)GO TO 1
      IF(ITEMP.EQ.2RID)GO TO 2
      IF(ITEMP.EQ.2RNO)GO TO 3
      PRINT*," UNRECOGNIZABLE INPUT INSPECTION BEING REACCOMPLISHED."
      GO TO 4
1     CONTINUE
      DO 10 J=1,4
      IACC(I,J)=0
10    CONTINUE
      RETURN NORM
```

AD-A055 224 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
XEROX CONTROLLER DESIGN.(U)
MAR 78 B W CORR
UNCLASSIFIED AFIT/GE/EE/78-11 NL

2 OF 2
AD
A055224

END
DATE
FILMED
7-78
DDC

```
2     CONTINUE
      PRINT 210
210   FORMAT(* ENTIRE ID INFORMATION MUST BE REENTERED UP*
     &/,* TO A MAX OF 30 CHARACTERS.*/* #*)
      READ 220,(IACC(I,J),J=1,3)
220   FORMAT(3A10)
      PRINT 230
230   FORMAT(* ENTER CLASS DESIGNATOR.*/* #*)
      READ 240,IACC(I,4)
240   FORMAT(A10)
      RETURN NORM
3     CONTINUE
      RETURN NORM
      END
```

C EPROM PROGRAMMING

The Imsai 8048CC single board computer is based on the 8035 microprocessor and the 2716 2kbyte EPROM for program memory, both from Intel. The 2716 (Ref 4) operates on a single 5 volt power supply and is programmable with TTL level signals after adding the 25 volt programming power supply. After erasure the 2716 contains all 1's and the desired locations must be programmed to 0's. The programming is accomplished by putting the desired address on the address lines, 1 on the $\overline{\text{CS}}$ line, and a 50 millisecond active high pulse on the PD/PGM line. If the 8035 microprocessor, the 8279 Keyboard Interface, and the two SN 74LS174's are removed the only connections to the address and data busses are the 2102AL RAM chips and the 2716 EPROM chip. External access to these lines is gained by using 16 pin connectors and flat cables plugged into the sockets vacated by the SN 74LS174's. A 10 bit counter is used to sequence the address lines (LA0-LA10) while the RAM is enabled in the read mode and the $\overline{\text{CS}}$ line on the 2716 is held high. The 50 millisecond pulse required on the PD/PGM line of the 2716 is produced by placing the pulse on the LA11 address line. Placing 1's on the $\overline{\text{WR}}$, $\overline{\text{RD}}$, and $\overline{\text{DSEN}}$ lines and a 0 on the $\overline{\text{RAMEN}}$ causes the data in the RAM at the address specified to be placed on the data lines and then into the 2716 when the pulse is placed on the LA11 line. The easiest way to put the 25 volts on the Vpp pin is to carefully bend it out and use an alligator clip to make the connection. The other connections discussed are made with

the use of IC clips and jumper cables. In this transfer technique the LA10 jumper on the 2716 must be manipulated to place the program to be transferred into the upper or lower segment of the program memory. In a similar manner the contents of the 2716 can be transferred directly into the RAM. By placing 0 on the LA11 line, the $\overline{\text{DSEN}}$ line, and pin 5 of U8 and the complement of the programming pulse used in the previous procedure on the $\overline{\text{WR}}$ line, the transfer is made from the 2716 to the RAM. For the purpose of the Xerox Controller program, the monitor which was supplied in the upper segment of the EPROM, was transferred into the RAM, the EPROM was erased, and the monitor replaced in the lower segment. The tested program (Xerox Controller program) was then placed in the RAM and transferred into the upper segment of the EPROM. The execution of the Xerox Controller program requires following the jumper notes for the high version of the Imp 48 contained in the 8048CC Control Computer User Manual (Ref 3). This rather straightforward procedure makes programming or modifying an existing program a simple operation.

D XEROX CONTROLLER

OPERATING INSTRUCTIONS

The Xerox Controller has two operating modes: user and control. The user mode is the normal operating mode, while the control is key selected. The two modes are described in the sections which follow.

User Mode

In order to access an account and enable the Xerox machine to make copies the user simply presses the reset (yellow) button and then enters his encrypted 6 hexadecimal digit user number. As each digit is entered 8. will appear on the display to indicate that the digit has been received. If at any time while entering the number the user feels that he has made an error or if the machine is not enabled after the entry of the sixth digit, the reset should be pressed and the procedure repeated. When a valid user number has been entered the present total will be displayed indicating that the machine is enabled and copies can be made. Each time a copy is made the account will be incremented by the copy cost, up to a total of 99 dollars and displayed on the display.

Control Mode

In order to enter this mode of operation the key holder inserts the key and turns it to the control position. The Controller will now respond to any of the 7 control mode commands described below. As these commands are entered

they are echoed on the display so they may be confirmed before the command to execute. The command to execute is given simply by pressing any one of the hexadecimal digits after the 8 digit command has been confirmed on the display. If the command on the display is incorrect, press the reset and reenter it.

<u>Control Mode Commands</u>

<u>Increment</u> 1AAAMMMM

AAA 3 digit hexadecimal account number

MMMM 4 digit decimal amount

This command increments the account (AAA) by the amount ($MM.MM). Completion of the command is indicated by the appearance of the account total on the display. If the increment would cause an overflow the display will go blank indicating the command has been rejected.

<u>Decrement</u> 2AAAMMMM

AAA 3 digit hexadecimal account number

MMMM 4 digit decimal amount

This command decrements the account specified (AAA) by the amount ($MM.MM). Completion of the command is indicated by the appearance of the account total on the display. If the decrement would cause an underflow (less than zero) the display will go blank indicating that the command has been rejected.

<u>Set</u> 3AAAMMMM

AAA 3 digit hexadecimal account number

MMMM 4 digit decimal amount

This command sets the account specified (AAA) to the amount ($MM.MM). Completion of the command is indicated by the appearance of the total on the display.

<u>Print</u> 4DDDDDDD

DDDDDDD any 7 digits

This command causes the printout of the hexadecimal account number and account total of any nonzero accounts. This printout can be produced on the RS 232C compatible TI Silent 700 (model 735) used with the external interface selected and the 300 baud speed specified. The connection is via the cable emanating from the back of the Xerox Controller.

<u>Clear</u> 5D505050

D any digit

This command causes all accounts in the controller to be set to 0000. In order to prevent an accidental clear command, the command must appear exactly as shown above. Completion is indicated by the appearance of zeros on the display.

<u>Code</u> 6DDDCCCC

DDD any 3 digits

CCCC 4 digit code provided by the Accounts Program

This command sets the encryption code in the controller which is used to decipher the user number to detect a valid account number. Completion of the command is indicated by the appearance of the code on the display.

Cost 7DDDDDCC

DDDDD any 5 digits

CC new copy cost in cents

This command sets the charge levied against an account when a copy is made. Completion of the command is indicated by the appearance of the cost on the display.

Maintenance Requirements

The Xerox Controller should be continously plugged in to 110 volts. The controller has a battery backup for the memory contents which will protect the memory contents for over 8 days. This battery is a Power Conversion Eternacell (model 550-6A). It is recommended that this battery be replaced once a year or if there is a long term power failure (over 24 hours). Trouble shooting malfunctions can be done using the design paper written by Capt Brian Corr as a thesis at AFIT.

E ACCOUNTS PROGRAM OPERATING PROCEDURES

This program produces and modifies a file containing the account number, user identification, class number, and user number (encrypted account number). Three files are involved in the operation of this program: tape1 (previously produced file), tape2 (new file), and tape3 (contains any printouts requested in the program run). To make changes or produce printouts of an existing file it must be available as a local file named tape1. Local files tape2 and tape3 will be produced by a program execution and therfore must not preexist. All the options which may be selected and the functions of the options are presented in detail below.

System Options: Create- Used to enter encryption coding information and user identification information for the production of a new file.

Update- Used to request printouts or make changes to a previously existing file (tape1 or just completed create).

Exit- Exits the program without accomplishing anything.

Create Options: The only option in the Create mode is to continue entering information or to terminate entry.

Update Options: Print- Prints the encryption code and corresponding decryption code to be used in the Xerox Controller. It also prints all or any class' accounts including account number, identification, class designation, and encrypted user number.

Code- Displays current encryption information and upon request assists in the selection of a new code which is then used to produce new user numbers for all the accounts

New- The New option is used to enter new users into the file. The first open space in the file is used.

Inspect- Displays and allows the deletion or change of user identification information and class designation.

End- Places the account information and encryption code information onto the file tape2 and then rewinds tapes 1, 2, and 3 in preparation for the program termination.

Vita

Brian William Corr was born 17 July 1947 in New York, New York. After graduation from Paramus High School in Paramus, New Jersey he attended Newark College of Engineering (renamed New Jersey Institute of Technology), from which he received a Bachelor of Electrical Engineering in June 1969. He was employed as an associate design engineer by Raytheon, Wayland Masschusetts until entering the Air Force in April 1971. He was commissioned, after completing OTS, in August 1971 and completed pilot training and received his wings in December 1972 at Columbus AFB, Mississippi. He served as a pilot and aircraft commander in the C-141 at Mcguire AFB, New Jersey until entering the School of Engineering, Air Force Institute of Technology in August 1976.

Permanent address: 3 Troast Rd

Paramus, NJ 07652

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

14

1. REPORT NUMBER AFIT/GE/EE/78-11	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) XEROX CONTROLLER DESIGN.		5. TYPE OF REPORT & PERIOD COVERED ~~MS Thesis~~ Master's thesis
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Brian W. Corr ~~Captain~~ [redacted]		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Air Force Institute of Technology (AFIR-EN) Wright-Patterson AFB, Ohio 45433		12. REPORT DATE Mar[redacted]78
		13. NUMBER OF PAGES ~~107~~ 106 p.
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

Approved for public release; IAW AFR 190-17
JERRAL F. GUESS, Captain, USAF
Director of Information

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Microcomputer
Accounting
Control
Computer Programs

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

A microprocessor system was designed, based on the Imsai 8048CC Single Board Computer (SBC), to identify authorized volume users of the Xerox copying machine operated by the AFIT Engineering Student Council. A power relay mounted on the SBC was used to enable the Xerox to make copies and a buffer circuit, based on the SN74121 one shot, was employed to count the copies made so that they could be charged to the identified user's account. An interactive FORTRAN program was implemented, on the CYBER system

available at AFIT, to keep a file of the authorized users, their unique user numbers, and the information necessary to bill them. The program in the microprocessor was designed to allow the treasurer of the Student Council to set the copy cost, the code used to identify valid user numbers, and any account total (increment, decrement, and set). Routines are also provided to clear all accounts and print all nonzero account totals on the TI model 735 terminal. The program is interrupt driven and the account totals, copy cost, and code maintained in memory are protected from loss by a battery backup circuit which was added to the SBC.